TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

VILLE SIRKIÄ

VÄRÄHTELYPERUSTAINEN VAURIONTUNNISTUSKOMPONENTTI JA MITTALAITEKOHTAINEN KALIBROINTI

Diplomityö

Tarkastaja: Prof. Tommi Mikkonen
Tarkastaja ja aihe hyväksytty
Tieto- ja sähkötekniikan
tiedekuntaneuvoston
kokouksessa 04.05.2016

TIIVISTELMÄ

VILLE SIRKIÄ: Värähtelyperustainen vauriontunnistuskomponentti ja mittalaitekohtainen kalibrointi
Tampereen teknillinen yliopisto
Diplomityö, 54 sivua
Kesäkuu 2016
Tietotekniikan koulutusohjelma
Pääaine: Ohjelmistotuotanto
Tarkastajat: Prof. Tommi Mikkonen
Avainsanat: vauriontunnistus, koneoppiminen, tukivektorikone, relevanssivektorikone

Kappaleissa ja rakenteissa olevia vaurioita voidaan havaita värähtelymittauksia analysoimalla. Kappaleen värähtelyä mitataan tyypillisesti herättämällä kappale ensin värähtelemään esimerkiksi herätevasaralla ja mittaamalla sitten herännyt värähtely kiihtyvyysantureilla. Vaurioiden tunnistamiseksi värähtelymittauksista voidaan hyödyntää erilaisia koneoppimisalgoritmeja, esimerkiksi tukivektorikoneita ja relevanssivektorikoneita.

Diplomityön taustalla on Atostek Oy:n asiakasyrityksen tarve ohjelmistokomponentille, joka kykenee havaitsemaan kappaleiden vaurioitumisen värähtelymittauksista sellaisten aikaisempien mittauksien perusteella, joissa mitatun kappaleen kunto tunnetaan.

Työssä valitaan tehtävään soveltuvat koneoppimisalgoritmit sekä suunnitellaan ja toteutetaan asiakasyrityksen isäntäjärjestelmään integroitava ohjelmistokomponentti, jolla vauriontunnistustehtävät suoritetaan. Lisäksi esitellään kalibrointimenetelmä, jolla eri mittalaitteilla ja mittaustavoilla mitattuja näytteitä voidaan käyttää koneoppimisalgoritmien opettamiseen.

Työn tuloksena syntyi vauriontunnistuskomponentti, jota voidaan käyttää yleiskäyttöisenä työkaluna kappaleiden kuntoluokitteluun niistä mitattujen värähtelymittausten perusteella. Saadun asiakaspalautteen perusteella komponentti täyttää asiakasyrityksen tarpeen sekä luokittelutarkkuuden että suorituskyvyn osalta. Esitetty kalibrointimenetelmä ei vielä tämän työn puitteissa tehdyn tutkimuksen pohjalta täytä kaikkia sille asetettuja odotuksia vaan vaatii jatkokehitystä toimiakseen tavoitellun mukaisesti.

Työn lopussa esitetään jatkokehitysajatuksia vauriontunnistuskomponentille. Näistä tärkeimmät ovat uusien työkalujen mahdollistama ohjelmiston osien välisen integraation parantaminen sekä luokittelun laajentaminen useamman kuin kahden luokan luokittelijoihin.

ABSTRACT

VILLE SIRKIÄ: Vibration-based damage detection component and measuring device specific calibration
Tampere University of Technology
Master's Thesis, 54 pages
June 2016
Master's Degree Programme in Information Technology
Major: Software engineering
Examiner: Prof. Tommi Mikkonen
Keywords: damage detection, machine learning, support vector machine, relevance vector machine

Damage in objects and structures can be detected by analyzing vibration measurements. Vibration is typically measured by exciting vibration in the object externally by using an impact hammer or other excitation device. The vibration is then measured using accelerometers. Various machine learning algorithms, e.g. support vector machines or relevance vector machines, can be applied to analyze the measurements and detect damage in the objects.

The background for this thesis is a client of Atostek Ltd. with a need for software component that detects damage from certain types of objects. The damage detection should be based on previous vibration measurements of objects whose condition is known.

In this thesis, suitable machine learning algorithms are chosen for the damage detection task. The design and implementation of the damage detection component using the chosen algorithms is then described. In addition, a calibration method for samples collected from various measurement instruments and with various measuring styles is also presented in this thesis.

The damage detection component was designed to be used as a universal classifier tool for vibration-based damage detection. Based on the feedback gathered from the client, the classification accuracy and performance of the implemented component fulfills client's needs. The proposed method for the measurement calibration did not reach all the goals in the scope of this thesis.

Some future development ideas are presented in the end of this thesis. The damage detection component could be further developed by improving the communication and integration between different parts of the software. Support for multi-class classifiers could be implemented to achieve more fine-grained classification.

ALKUSANAT

Tämä diplomityö on tehty Atostek Oy:ssä ja siinä tehty tutkimus liittyy erään Atostekin asiakasyrityksen projektiin. Haluan esittää kiitokset Atostekille diplomityön rahoittamisesta sekä ajan järjestämisestä sen kirjoittamiseksi muiden töiden ohessa. Asiakasyritystä haluan kiittää mielenkiintoisen tutkimusaiheen tarjoamisesta.

Haluan kiittää myös työn tarkastajaa professori Tommi Mikkosta sekä työn ohjaajaa Mika Torholaa, joilta sain hyvää ohjausta, arvokasta palautetta sekä uusia näkökulmia työn sisältöön. Erityiskiitos kuuluu vanhemmilleni, avovaimolleni sekä tädilleni, jotka ovat väsymättä kannustaneet minua eteenpäin niin diplomityön kirjoittamisessa kuin muissakin opinnoissani.

Tampere, 12.05.2016

Ville Sirkiä
ville.sirkia@iki.fi

SISÄLLYS

KUVALUETTELO

LYHENTEET JA MERKINNÄT

CSD	Cross spectral density
MATLAB	MATrix LABoratory. Numeeriseen laskentaan tarkoitettu laskentaympäristö sekä ohjelmointikieli.
NDT	Non-destructive testing, ainetta rikkomaton vauriontunnistus
PRT	Pattern recognition toolbox -kirjasto MATLABille [1]
PSD	Power spectral density
RVM	Relevance vector machine, relevanssivektorikone
SLDV	Scanning laser Doppler vibrometer, laseriin perustuva värähtelyä mittaava laite
SNR	Signal to noise ratio, signaali-kohinasuhde
SVM	Support vector machine, tukivektorikone

1. JOHDANTO

Erilaisten kappaleiden ja rakenteiden kunnonvalvonta on perinteisesti perustunut silmämääräiseen tarkasteluun. Silmämääräisen tarkastuksen lopputulos voi olla riippuvainen tarkastajan tulkinnasta, eikä tarkastuskohde yleensä ole kokonaisuudessaan silmämääräisen tarkastelun ulottuvissa, jolloin tarkastuksen kattavuus voi jäädä riittämättömäksi. Lisäksi tällaisella menetelmällä kappaleen jäljellä olevaa käyttöikää on ollut vaikea arvioida luotettavasti.

Tällaisten tarkastuksen kattavuutta rajoittavien tekijöiden poistamiseksi on kehitetty erilaisia ainetta rikkomattomia vauriontunnistusmenetelmiä, joilla kappaleiden rakenteellista eheyttä voidaan tutkia silmämääräistä tarkastelua tarkemmin. Eräs tällainen menetelmä on värähtelyperustainen vauriontunnistus. Tässä menetelmässä analysoitava kappale saatetaan ensin värähtelemään ja herännyt värähtely mitataan. Mitattua värähtelyä analysoimalla voidaan määrittää, onko kappale ehjä vai vaurioitunut. Myös vaurion sijainti kappaleessa on mahdollista määrittää värähtelymittauksen perusteella.

Eräällä Atostek Oy:n asiakasyrityksellä on tarve ohjelmistokomponentille, joka kykenee kappaleesta tehtyjen värähtelymittauksien perusteella tunnistamaan ehjät kappaleet vaurioituneista. Tavoitteena on kehittää ohjelmisto, joka voidaan opettaa luokittelemaan kunnoltaan tuntemattomia näytteitä sellaisten näytteiden pohjalta, joiden kunto tunnetaan ennestään. Koneoppimisen termein vauriontunnistuskomponentin tulee olla ohjattuun oppimiseen perustuva luokittelija. Ohjatulla oppimisella tarkoitetaan oppimistapaa, jossa koneoppimisalgoritmi opetetaan sellaisilla näytteillä, joiden luokat tunnetaan entuudestaan. Luokittelijaa opetettaessa opetusnäytteistä ja niihin liittyvistä luokista muodostetaan luokittelijamalli, jota käyttäen tuntemattomia näytteitä voidaan luokitella.

Värähtelymittausten analysoimiseksi ja luokittelemiseksi tässä diplomityössä suunnitellaan ja toteutetaan vauriontunnistuskomponentti. Komponenttia käytetään asiakasyrityksen Microsoft .NET:llä toteutetusta isäntäjärjestelmästä, joten vauriontunnistuskomponentin on tarjottava .NET-yhteensopiva rajapinta.

Asiakasyrityksen käytössä on erilaisia mittauslaitteita ja -tapoja, joilla luokiteltavia kappaleita mitataan. Tässä työssä esitellään tapa, jolla eri mittalaitteilla ja mittaustavoilla mitatut näytteet voidaan kalibroida keskenään vertailukelpoisiksi. Kalibroinnin tavoitteena on, että useilla eri mittauslaitteilla ja -tavoilla mitattuja näytteitä voidaan käyttää vauriontunnistuskomponentin opetusnäytteinä.

Rakenteellisesti työ jakautuu seitsemään lukuun. Luvun 2 aiheena on värähtelyperustainen vauriontunnistus. Luvussa esitellään millaisia työkaluja käyttäen kappale voidaan herättää värähtelemään, miten värähtelyä voidaan mitata ja millaisia vaurioita värähtelymittausten perusteella voidaan havaita. Luvussa 3 tutustutaan koneoppimiseen. Siinä esitellään koneoppimisen taustaa sekä yleisimpiä koneoppimisen sovelluksia ja kuvataan vauriontunnistuskomponenttiin soveltuvien koneoppimisalgoritmien valinta. Luvussa käsitellään myös koneoppimisalgoritmien syötteenä käytettävien piirteiden valintamenetelmiä. Lisäksi esitellään tarkemmin tukivektorikone ja relevanssivektorikone sekä vertaillaan niiden ominaisuuksia. Luku 4 keskittyy vauriontunnistuskomponenttiin. Luvussa kuvataan isäntäjärjestelmälle tarjottavan rajapinnan toiminnot, vauriontunnistuskomponentin arkkitehtuuri sekä .NET-kääreen ja MATLAB-komponenttien toiminta. Lisäksi käsitellään komponenttien välistä kommunikointia ja virheenkäsittelyä. Luvussa 5 kuvataan menetelmä, jolla eri mittauslaitteilla ja -tavoilla mitattuja näytteitä voidaan kalibroida keskenään vertailukelpoisiksi. Lisäksi arvioidaan kalibroitujen näytteiden käyttökelpoisuutta vauriontunnistuskomponentin kanssa. Luvussa 6 arvioidaan vauriontunnistuskomponentin ja mittalaitekohtaisen kalibroinnin soveltuvuutta tehtäviinsä sekä esitetään jatkokehitysajatuksia työn tuloksena syntyneiden komponenttien kehittämiseksi tulevaisuudessa. Luku 7 sisältää yhteenvedon tässä työssä saavutetuista tuloksista.

2. VÄRÄHTELYPERUSTAINEN VAURIONTUNNISTUS

Perinteisesti erilaisten kappaleiden ja rakenteiden kunnonvalvonta on perustunut silmämääräiseen tarkasteluun. Tällainen tarkastaminen säännöllisesti tehtynä on kuitenkin verraten kallista ja monissa tarkastuskohteissa myös epätarkkaa, sillä osa kohteesta saattaa olla silmämääräisen tarkastuksen ulottumattomissa. Lisäksi silmämääräisellä tarkastamisella ei usein voida arvioida luotettavasti tarkastuskohteen jäljellä olevaa kestävyyttä kvantitatiivisesti, ja tarkastuksen lopputulos voi olla riippuvainen tarkastajan tulkinnasta. [2]

Kunnonvalvonnan tarkkuuden ja kustannustehokkuuden parantamiseksi on kehitetty rikkomattomaan aineenkoetukseen (non-destructive testing, NDT) perustuvia vauriontunnistusmenetelmiä [3]. Yhteistä näille menetelmille on se, että tarkastus ei aiheuta pysyvää vauriota kohteeseen. Tällaiset menetelmät voivat perustua esimerkiksi röntgenkuvaukseen, ultraääneen tai kappaleen värähtelyominaisuuksiin, johon tässä luvussa käsiteltävä värähtelyperustainen vauriontunnistus pohjautuu.

Tässä luvussa esitellään värähtelyperustaiseen vauriontunnistukseen liittyvät oleellisimmat vaiheet, eli värähtelyn ja herätteen mittaaminen, taajuusvastefunktion muodostaminen mittauksista sekä vaurioiden tunnistaminen taajuusvastefunktion pohjalta tehtävän analyysin perusteella.

2.1 Värähtelyn mittaaminen

Värähtelyperustaisessa vauriontunnistuksessa tavoitteena on havaita kappaleessa ilmenneiden vaurioiden aiheuttamia muutoksia kappaleen värähtelyominaisuuksia tutkimalla. Värähtely voi olla lähtöisin joko kappaleesta itsestään tai ulkopuolisesta herätteestä. Kappale voi värähdellä itsestään esimerkiksi silloin kun mittauskohteena on käynnissä oleva kone ja värähtelyn aiheuttaa sen moottori. Jos kappale ei värähtele itsestään, värähtelyn mittausta varten kappale saatetaan värähtelevään tilaan käyttämällä värähtelyn herätteenä esimerkiksi herätevasaraa tai elektromagneettista täristintä [5, s.18]. [4, s.1]

Herätevasara (kuvassa 2.1 vasemmalla) muistuttaa ulkonäöltään tavallista vasaraa. Vasaran lyöntiosa muodostuu lyöntipäästä, voima-anturista ja tasapainomassasta. Herätesignaalin lähteenä herätevasara on paljon käytetty vaihtoehto, koska vasaralla mittaaminen on nopeaa ja lisäksi mittaamiseen tarvitaan vain vähän kalustoa, joka on myös verraten edullista hinnaltaan. Herätevasaralla saadaan tyypillisesti herätettyä ominaistaajuuksia laajemmalla taajuuskaistalla kuin täristimillä. Herätevasara on käsikäyttöinen mittalaite, jonka takia tasalaatuisten mittausten tuottaminen voi olla haastavaa. [5, s.22]

Kuva 2.1 *Brüel & Kjaerin [6] värähtelymittaustuotteita: herätevasara, elektromagneettinen täristin ja kiihtyvyysanturi*

Elektromagneettisissa täristimissä (keskellä kuvassa 2.1) värähtely syntyy, kun syötteenä saatu sinimuotoinen signaali muutetaan sähkökentän vaihteluiksi, joilla ohjataan värähtelevää magneettista kelaa. Värähtely siirretään mitattavaan kappaleeseen yhdystangolla, jotta täristimen massa ei vaikuttaisi kappaleen massaan ja muuttaisi näin sen värähtelyominaisuuksia. Värähtelyn taajuutta ja muotoa säädellään signaaligeneraattorilla ja herätevoiman suuruutta vahvistimella. Signaaligeneraattori ja vahvistin ovat joissain täristinmalleissa myös sisäänrakennettuina.[5, s.19]

Kappaleen värähtely mitataan esimerkiksi siihen kiinnitetyillä kiihtyvyysantureilla (oikealla kuvassa 2.1), joilla värähtely saadaan tallennettua ajan funktiona. Kappaleen värähtelyn lisäksi myös herätesignaali mitataan, jotta saadaan muodostettua kappaleen värähtelyn ja herätteen välistä suhdetta kuvaava taajuusvastefunktio. Herätesignaali mitataan herätteen tuottavassa laitteessa olevan voima-anturin avulla. [5, s.15]

Iskumuotoinen vasaraheräte on kestoltaan hyvin lyhyt, ja sen signaali-kohinasuhde (signal to noise ratio, SNR) on tyypillisesti huono. Kohinan vähentämiseksi heräte-

signaali voidaan ikkunoida force-ikkunalla, jonka tarkoitus on parantaa signaalin ja kohinan suhdetta poistamalla signaalista kohinaa niiltä osin kun herätevasara ei ole kosketuksissa herätettävän kappaleen kanssa. [7]

Kiihtyvyysantureiden ja muiden kappaleeseen suoraan kiinnittyvien laitteiden lisäksi värähtelyn mittaamiseen voidaan käyttää myös suhteellisen uutta laserin heijastumiseen perustuvaa SLDV-tekniikkaa (scanning laser Doppler vibrometer), jolla mittaus voidaan suorittaa kosketuksettomasti ja suurella tarkkuudella. Tällä tekniikalla värähtelyn taajuus voidaan havaita jopa 250 kilohertsiin asti ja amplitudi nanometrien tarkkuudella.[8, s.579]

2.2 Taajuusvastefunktio

Ajan funktiona esitetty signaali muunnetaan taajuusavaruuteen käyttäen Fourier-muunnosta. Aika-avaruuden signaali jaetaan muunnoksessa sinimuotoisiin taajuuskomponentteihin, joista lasketaan integraali. Taajuuskomponenttien summista muodostuu taajuusspektri. Jatkuvassa Fourier-muunnoksessa ajan funktio tulee kuitenkin olla jatkuva ja käytännön sovelluksissa mittalaitteilla ei pystytä mittaamaan äärettömän tiheitä mittauksia, mutta mittauspisteet ovat tyypillisesti tasavälisiä. Tällöin taajuusavaruuden esityksen laskemiseksi käytetään diskreettiä Fourier-muunnosta, jossa kullekin mittauspisteelle lasketaan taajuuskomponentti. Taajuuskomponenttien summa muodostaa taajuusspektrin. Diskreetti Fourier-muunnos voidaan laskea tehokkaasti Fast Fourier Transform -algoritmin (FFT) [10] avulla. [9, s.190-194]

Taajuusvastefunktio on kompleksimuotoinen funktio, joka kuvaa herätteen ja vasteen suhdetta värähtelevässä systeemissä. Perustapauksessa taajuusvastefunktio voidaan laskea herätteen ja vasteen FFT-muunnosten pohjalta kaavalla

$$H(f) = \frac{Y(f)}{X(f)}, \tag{2.1}$$

jossa $H(f)$ on systeemin taajuusvastefunktio, $Y(f)$ on vastesignaalin FFT-muunnos ja $X(f)$ on herätesignaalin FFT-muunnos. [9, s.139]

Koska vauriontunnistuskomponentin tapauksessa herätesignaali on tyypillisesti hyvin lyhyt, herätteessä oleva kohina poistetaan ikkunoimalla. Tällöin on odotettavissa, että vasteessa on paljon kohinaa herätteeseen verrattuna. Tällöin taajuusvastefunktion funktio voidaan laskea CSD:n (cross spectral density) ja PSD:n (power spectral density) suhteesta [9, s.684]. CSD muodostetaan herätteestä ja vasteesta

kaavalla

$$P_{XY}(f) = CSD(a,b)(f) = \frac{FFT^*(X)(f)FFT(Y)(f)}{N^2}, \tag{2.2}$$

jossa N on näytteiden määrä ja $FFT^*(X)$ herätesignaalin taajuustason esityksen kompleksikonjugaatti. PSD muodostetaan herätteelle kaavalla

$$P_{XX}(f) = PSD(X)(f) = CSD(X,X)(f). \tag{2.3}$$

Taajuusvastefunktio (FRF) lasketaan nyt CSD:n 2.2 ja PSD:n 2.3 suhteesta

$$FRF(f) = \frac{P_{XY}(f)}{P_{XX}(f)} = \frac{FFT^*(X)FFT(Y)}{FFT^*(X)FFT(X)}. \tag{2.4}$$

2.3 Vaurioiden tunnistaminen

Värähtelyperustaisessa vauriontunnistuksessa tutkitaan kappaleen ominaistaajuuksissa, ominaismuodoissa sekä vaimenemiskertoimissa tapahtuvia muutoksia. Näitä *modaaliparametreiksi* kutsuttuja arvoja ei voida laskea suoraan kokeellisesti määritetystä taajuusvastefunktiosta useita vapausasteita (MDOF, multiple degrees of freedom) käsittävässä systeemissä, vaan ominaistaajuuksien päällekkäisyyden vuoksi taajuusvastefunktio on muutettava analyyttiseksi siirtofunktioksi. [9, s.737]

Taajuusvastefunktio muutetaan analyyttiseksi siirtofunktioksi sopivaa käyränsovitusmenetelmää (*curve fitting*) käyttäen. Modaaliparametrit kappaleen ominaistaajuuksille saadaan tämän jälkeen laskettua siirtofunktiosta. Käyränsovitusmenetelmiin ei paneuduta tarkemmin tässä työssä, mutta mainittakoon, että tarkoitukseen voidaan käyttää useita eri käyränsovitusmenetelmiä, esimerkiksi Rational Fraction Polynomial -menetelmää (RFP) [11] tai Pronyn menetelmää [12]. [9, s.737]

Käyränsovitusmenetelmien lisäksi modaaliparametrit voidaan määrittää myös ns. “peak picking”-menetelmällä, jossa modaaliparametrit määritetään suoraan taajuusvastefunktiosta. Ominaistaajuudet näkyvät taajuusvastefunktiossa piikkeinä ja kukin ominaistaajuus kuvaa yhtä kappaleen värähtelemään herännyttä vapausastetta. Peak picking -menetelmässä jokaiseen taajuusvastefunktiossa näkyvään ominaistaajuuteen sovitetaan analyyttisesti siirtofunktio, josta saadaan laskettua modaaliparametrit. Tämä menetelmä toimii vain sellaisissa tapauksissa, kun ominaistaajuudet ovat taajuusvastefunktiossa selvästi erillään toisistaan ja niiden vaimennuskerroin on pieni. [9, s.742]

Mannan et al. [13] mainitsevat tutkimuksessaan, että modaaliparametrien muutosten perusteella voidaan havaita useita erilaisia vikoja tutkittavassa kappaleessa. Täl-

laisia ovat materiaaliviat (esim. murtuminen tai katkeaminen), toisiinsa kiinnitettyjen osien liitoksien heikkeneminen (ruuvien löystyminen, liimasaumojen pettäminen), monet valmistusviat sekä kokoonpanossa tapahtuneet virheet. Yleisemmin ilmaistuna modaaliparametrien avulla voidaan siis tunnistaa muutoksia tutkittavan kappaleen massassa, jäykkyydessä tai vaimennusominaisuuksissa.

Vaurioiden vaikutus testattavan kappaleen modaaliparametreissa riippuu aina kappaleen ominaisuuksista, jolloin ei voida yleispätevästi määritellä, millaisia muutoksia modaaliparametreihin aiheutuu tietynlaisten vaurioiden ilmetessä. Tässä työssä toteutettavan vauriontunnistuskomponentin tavoitteena on kyetä koneoppimisen keinoin havaitsemaan vaurioiden vaikutukset mitatun kappaleen värähtelyominaisuuksissa ja pystyä tähän perustuen ilmaisemaan, onko mitattava kappale rikkinäinen vai ehjä.

3. KONEOPPIMINEN

Koneoppimisella tarkoitetaan aikaisempaa kokemusta hyödyntäviä laskennallisia menetelmiä, joiden avulla saadaan tuotettua ennusteita ennalta tuntemattomista näytteistä. Aikaisemmalla kokemuksella tarkoitetaan koneoppijan käytettävissä olevia tunnettuja näytteitä samankaltaisesta ongelmasta. Näytteet ovat ongelmaan liittyvää tietoa, jota käytetään koneoppimisalgoritmin syötteenä sekä algoritmia opetettaessa että ennusteita tuotettaessa. Tyypillisesti koneoppimisalgoritmien opetusvaiheessa muodostetaan tunnettujen näytteiden perusteella malli, jota käyttäen tuntemattomista näytteistä voidaan tuottaa ennusteita. [14]

Tässä luvussa esitellään tyypillisiä koneoppimisen avulla ratkaistavia ongelmia ja niihin soveltuvia algoritmeja sekä valitaan toteutettavaan vauriontunnistuskomponenttiin soveltuva luokittelualgoritmi. Lisäksi perehdytään koneoppimisalgoritmille syötettävien piirteiden valintaan sekä tarkastellaan ja vertaillaan vauriontunnistuskomponentissa tarvittaviin luokittelutehtäviin soveltuvia algoritmeja.

3.1 Koneoppimisalgoritmit

Yleisiä koneoppimisen sovelluksia ovat mm. tekstin ja dokumenttien luokittelu, konenäkö, puheentunnistus ja monet lääketieteen diagnooseihin liittyvät sovellukset. Mohrin [14, s.2] mukaan tunnetuimmat koneoppimisen avulla selvitettävät ongelmat, joihin monet edellä mainituista sovelluksistakin perustuvat, ovat seuraavat:

- **Luokittelu:** Näytteiden luokittelu ennalta määritettyihin kategorioihin niiden ominaisuuksien perusteella. Esimerkiksi tässä työssä toteutettava vauriontunnistuskomponentti luokittelee mittauskohteet niistä kerättyjen näytteiden perusteella rikkinäisiin ja ehjiin.
- **Regressio:** Käyttäytymisen, tapahtuman tai toiminnan ennakointi koneoppimismallin avulla, esimerkiksi pörssikurssien ennustaminen.
- **Arvostus:** Näytteiden lajittelu paremmuusjärjestykseen määrättyjen kriteerien mukaan.

- **Klusterointi:** Näytteiden ositus homogeenisiin ryhmiin tai alueisiin silloin kun ryhmiä ei ole ennalta määritetty. Klusterointia sovelletaan tyypillisesti suuriin datamääriin, kuten sosiaalisen verkon käyttäjiin liittyvään dataan. Klusteroimalla tätä dataa voidaan esimerkiksi analysoida käyttäjien jakautumista erilaisiin yhteisöihin.
- **Datan ulottuvuuksien määrän vähentäminen:** Alkuperäisen näytteen muuntaminen vähemmän ulottuvuuksia sisältävään esitystapaan samalla säilyttäen alkuperäisen esitysmuodon ominaisuuksia. Esimerkiksi digitaalisen kuvan väri-informaation poistaminen tai suppeampaan väriavaruuteen siirtäminen.

Koneoppimisalgoritmit voidaan ryhmitellä myös niiden oppimistavan mukaan. Bishop [15, s.3] jaottelee koneoppimisalgoritmit ohjattuun ja ohjaamattomaan oppimiseen sekä vahvistusoppimiseen (*reinforcement learning*) perustuviin algoritmeihin. *Ohjatussa oppimisessa* koneoppimisalgoritmin opettamisessa tunnettujen näytteiden lisäksi syötteenä on myös näytteisiin liittyvät kohdearvot, joilla algoritmille ilmaistaan, mikä tulos algoritmin odotetaan tuottavan kyseisellä näytteellä [15, s.21]. Näytteiden ja kohdearvojen perusteella muodostetaan malli, jonka pohjalta ennalta tuntemattomista näytteistä muodostetaan kohdearvoja. Luokittelu- ja regressioalgoritmit perustuvat tyypillisesti ohjattuun oppimiseen. Ohjattuun oppimiseen soveltuvia algoritmeja ovat mm. tukivektorikone [16], relevanssivektorikone [17], KNN (k-nearest neighbors) [15, s.125] sekä neuroverkkolaskentaan perustuva monikerrosperseptroni [18, s.179].

Ohjaamattomaan oppimiseen perustuvissa koneoppimisalgoritmeissa opetusnäytteistä odotettavat tulokset ovat algoritmille tuntemattomia. Algoritmi analysoi sille syötettyjä piirteitä, ja niistä havaittujen samankaltaisuuksien ja niiden välisten yhteyksien perusteella muodostetaan malli [15, s.3]. Ohjaamattomaan oppimiseen perustuvia algoritmeja voidaan käyttää mm. syötteiden klusterointiin, tiheysestimaattien muodostamiseen sekä datan ulottuvuuksien määrän vähentämiseen. Ohjaamattomaan oppimiseen perustuvia algoritmeja ovat mm. yksiluokkainen tukivektorikone [15, s.339], itseorganisoituva kartta (SOM) [19] sekä K-means -klusterointi [15, s.424].

Vahvistusoppimiseen perustuvissa algoritmeissa tavoitteena on löytää sopivimmat toimenpiteet annetussa tilanteessa. Toimenpiteiden sopivuuden mittarina käytetään niistä saatua palautetta, joka voi olla positiivista tai negatiivista. Saadun palautteen avulla algoritmi oppii, mitkä toimenpiteet eri tilanteissa eniten positiivista palautetta ja toimii tämän mukaisesti. Ohjattuun oppimiseen verrattuna vahvistusoppimisessa algoritmin käytettävissä ei ole esimerkkejä optimaalisista tuloksista, vaan op-

timaalinen käyttäytyminen opitaan ”yrityksen ja erehdyksen” kautta. Vahvistusoppiminen perustuu Markovin päätösprosessien (MDP) optimointiin, ja tähän tehtävään yleisesti käytettyjä työkaluja ovat mm. dynaaminen ohjelmointi, Monte Carlo -menetelmä sekä ns. TD-algoritmi (temporal difference).[20]

3.2 Vauriontunnistuskomponenttiin soveltuvat algoritmit

Tässä työssä toteutettavan vauriontunnistuskomponentin avulla tulee pystyä tunnistamaan toisistaan ehjät ja rikkinäiset mittauskohteet. Koneoppimisalgoritmin ratkaistavana tulee siis olemaan luokittelu- tai klusterointiongelma, jolloin algoritmin valinta rajataan tällaisia koneoppimisongelmia ratkaiseviin algoritmeihin.

Keskeisenä vaatimuksena komponentin suunnittelussa on, että algoritmin käyttäytyminen tulee voida määrittää opettamalla se värähtelymittauksista saadulla datalla, joka on mitattu sellaisista mittauskohteista, joiden kunto tunnetaan. Tämä vaatimus rajaa algoritmin valinnan ohjattuun oppimiseen perustuviin algoritmeihin. Klusterointia tekevät koneoppimisalgoritmit perustuvat ohjaamattomaan oppimiseen, joten ne rajautuvat valinnan ulkopuolelle. Komponentissa käytettävä algoritmi valitaan siis ohjattuun oppimiseen perustuvien luokittelualgoritmien joukosta.

Neuroverkot ovat koneoppimisalgoritmien ryhmä, joiden kehittämisen lähtökohtana on alun perin ollut biologisen hermoston mallintaminen matematiikan keinoin. Niiden yhteyttä biologisiin hermoverkkoihin usein liioitellaan, ja keinotekoiset neuroverkot tulisikin ennemmin nähdä lähinnä tehokkaina tilastollisina algoritmeina kuin inhimillistä oppimista vastaavana koneoppimismenetelmänä. [15, s.226]

Neuroverkkoalgoritmeista *monikerrosperseptroni* (MLP) täyttää vauriontunnistuskomponenttiin valittavalle algoritmille asetetut vaatimukset. Lisäksi käytännön sovellutuksissa MLP on osoittautunut yhdeksi käyttökelpoisimmista neuroverkkoalgoritmeista [15, s.226].

MLP koostuu vähintään kolmesta kerroksesta: *syötekerroksesta*, yhdestä tai useammasta *piilokerroksesta* sekä *ulostulokerroksesta*. Piilokerrosten määrä vaihtelee luokitteluongelman kompleksisuuden mukaan. Jokaisessa kerroksessa on yksi tai useampia laskentayksiköitä eli *neuroneita*, jotka koostuvat synapseista, summaajasta, aktivaatiofunktiosta sekä bias-arvosta. Synapsilla tarkoitetaan kahden neuronin välistä kytkentää, johon liittyy painokerroin. Summaaja puolestaan laskee syötesynapseilta saaduista arvoista painotetun summan, johon lisätään bias-arvo. Tästä välisummasta epälineaarinen aktivaatiofunktio tuottaa neuronin ulostulon. [18, s.178-179]

Neuroverkkoa opetettaessa neuronien väliset painokertoimet määritetään siten, että

kullakin opetusjoukon syötteellä ulostulokerros tuottaa mahdollisimman monissa tapauksissa oikean luokittelutuloksen. Optimaalisten painokerrointen saavuttamiseksi käytetään ns. *error backpropagation -algoritmia*, jossa neuroverkon tuottama virhe syötetään takaisin verkolle ja muutetaan verkon painokerrointen arvoja virheen pienentämiseksi. [15, s.243]

Vaikka monikerrosperseptroni täyttää tässä työssä toteutettavalle luokittelijalle asetetut perusvaatimukset, sitä ei kuitenkaan valittu käytettäväksi vauriontunnistuskomponentissa vielä tässä vaiheessa. Monikerrosrrosperseptroni on nopea opettaa, mutta se on tukivektorikonetta hitaampi ennusteiden tuottamisnopeudessa, sillä MLP suorittaa enemmän laskentaa ennusteiden tuottamisen yhteydessä. Vauriontunnistuskomponentin käyttöympäristössä luokittelijan opettamisnopeudella ei ole merkitystä, mutta luokittelutulosten muodostamisen tulee tapahtua mahdollisimman nopeasti.

Tukivektorikoneen valintaa puoltaa myös sen parempi luokittelutarkkuus tietyissä sovelluksissa. Friaz-Martinez et. al. [21] vertailivat tukivektorikoneen ja monikerrosperseptronin soveltuvuutta käsinkirjoitetun allekirjoituksen tunnistamisessa. Tässä vertailussa tukivektorikoneen luokittelutarkkuus oli monikerrosperseptronia parempi ja myös tukivektorikoneen yleistämiskyky todettiin merkittävästi ylisovittumiseen taipuvaista MLP:tä paremmaksi. Algoritmien soveltuvuus erilaisiin sovelluksiin voi vaihdella, mutta edellä esitettyjen seikkojen nojalla MLP jätetään tässä vaiheessa vauriontunnistuskomponenttiin sisällytettävien algoritmien joukosta pois.

Tässä työssä keskitytään kahteen koneoppimisalgoritmiin: kohdassa 3.5 esiteltävään tukivektorikoneeseen sekä joitain tukivektorikoneen puutteita korjaamaan kehitettyyn relevanssivektorikoneeseen, joka esitellään kohdassa 3.6. Molemmat ovat ohjattuun oppimiseen perustuvia algoritmeja, joilla on saatu hyviä tuloksia erilaisissa luokittelu- ja regressio-ongelmissa. Tukivektorikoneella saavutettiin esimerkiksi MNIST-kuvantunnistustietokannan suorituskykytestissä maailmanennätys vuonna 2002 [22]. Lisäksi sitä on käytetty menestyksellä muun muassa monissa eri lääketieteellisissä sovelluksissa [23]. Relevanssivektorikonetta puolestaan on käytetty hyvin tuloksin esimerkiksi värähtelyperustaisessa öljyhiekkapumpun kunnon tarkkailussa [24] sekä reaaliaikaisessa tietoverkkoihin kohdistuvassa tunkeutumisen havaitsemisessa [25].

3.3 Piirteiden valinta

Koneoppimisalgoritmien syötteenä käytettävät näytteet ovat koneoppimisongelman sovellusalueeseen liittyvää tietoa koneluettavassa muodossa. Yksittäiseen näyttee-

seen liittyy vektori, jonka alkioita kutsutaan *piirteiksi*. Piirteiden tavoitteena on kuvata koneoppimisongelmalle relevantteja ominaisuuksia, jotka erottavat näytteet toisistaan ongelman tavoitellun ratkaisun mukaisesti. Esimerkiksi luokitteluongelmissa optimaalinen piirre tai piirteiden yhdistelmä on sellainen, jonka arvot jakautuvat näytteiden luokkien mukaisesti. Ohjattuun oppimiseen perustuvissa algoritmeissa sekä opetusnäytteistä että tuntemattomista analysoitavista näytteistä lasketaan sama joukko piirteitä.

Tärkeimpänä koneoppimiseen vaikuttavana onnistumistekijänä Hall [26] mainitsee käytettyjen opetusnäytteiden esitystavan ja laadun. Jos opetusnäytteet sisältävät paljon epäoleellista tai toistuvaa tietoa, tai jos näytteet ovat kohinaisia tai epäluotettavia, koneoppimisalgoritmien on vaikea löytää niistä säännönmukaisuuksia ja oleellista tietoa. Lisäksi saatavilla olevan tiedon määrä kasvaa koko ajan, jolloin piirteiden valinnan merkitys korostuu entisestään [27, s.3]. Epäoleellisen tiedon tunnistaminen ja poistaminen vähentää piirrevektorien ulottuvuuksien määrää ja parantaa siten myös laskennan nopeutta ja tehokkuutta.

Piirteidenvalintamenetelmissä on kaksi keskeistä osaa: piirteiden osajoukon haku ja valitun osajoukon arviointi [27, s.23]. Piirteiden osajoukon hakuun voidaan käyttää esimerkiksi geneettisiä algoritmeja tai *simuloitua jäähdytystä* (simulated annealing). Valitun osajoukon arvioimisessa hyödynnetään piirteidenvalintamenetelmästä riippuen joko tilastollisia työkaluja tai valitun koneoppimisalgoritmin tuottamaa tulosta muodostetulla osajoukolla. Piirteidenvalintamenetelmät voidaan jaotella sen mukaan, missä vaiheessa piirteet valitaan koneoppimisalgoritmin suorittamiseen nähden.

Suodattava piirteidenvalinta on menetelmä, jossa optimaalinen piirteiden joukko valitaan ennen oppimisalgoritmin suorittamista ainoastaan piirrevektorien sisältämään dataan perustuen. Piirteiden valinnan jälkeen koneoppimisalgoritmi opetetaan parhaaksi osoittautuneella piirteiden osajoukolla, jonka jälkeen piirrejoukon optimaalisuutta voidaan arvioida. Suodattava piirteidenvalinta tutkii piirteiden relevanssia tilastollisia menetelmiä käyttäen, jolloin piirrejoukon ylisovittumisen riski pienenee. Suodattava piirteidenvalinta on muihin piirteidenvalintamalleihin verrattuna laskennallisesti nopeaa, sillä piirteidenvalinta suoritetaan oppimisalgoritmista riippumattomasti eikä potentiaalisesti raskasta koneoppijan opettamista tarvitse suorittaa [28, s.162]. Toisaalta samasta syystä korkeimman relevanssin piirteet eivät aina ole piirrejoukon käyttökelpoisimpia valitun koneoppimisalgoritmin kanssa käytettynä.[27, s.27]

Ohjatun oppimisen tapauksessa optimaalisen osajoukon valintaan voidaan suodat-

tavassa piirteenvalinnassa käyttää esimerkiksi Relief-perheen (Relief, RRelief, ReliefF ja RReliefF) piirteenvalinta-algoritmeja. Osajoukon hyvyys määritetään tällöin tutkimalla *luokkien välistä erottuvuutta* (class-based separation). Perustapaus Relief huomioi piirteiden numeeristen arvojen erojen lisäksi myös piirteiden välisen euklidisen etäisyyden piirreavaruudessa. Piirreavaruudessa toisiaan lähekkäin sijaitsevat piirteet ovat Relief-algoritmissa keskenään samankaltaisempia kuin kauempana toisistaan sijaitsevat piirteet [27, s.172]. Relief soveltuu kuitenkin ainoastaan kaksiluokkaisten luokitteluongelmien piirteenvalintaan. Lisäksi se on herkkä kohinalle eikä toimi ollenkaan jos piirrevektoreissa on puuttuvia arvoja.

Näiden rajoitteiden välttämiseksi kehitettiin ReliefF-algoritmi, joka toimii myös silloin kun piirrevektorista puuttuu arvoja. Tässä parannellussa algoritmissa myös luokkien määrä on rajoittamaton, sillä sen sijaan että piirteiden välistä etäisyyttä arvioitaisiin koko piirreavaruudessa, ReliefF arvioi yksittäisen luokan sisäisten piirteiden välistä etäisyyttä. Lisäksi herkkyyttä kohinalle on vähennetty huomattavasti laskemalla piirteen relevanssi keskiarvottamalla piirrevektorien arvoja yksittäiselle piirteelle Reliefin yksittäisen piirrevektorin käyttämisen sijaan. [27, s.175] Relief-algoritmeista on tässä esiteltyjen perustapausten lisäksi olemassa myös tietyille sovellusalueille tehtyjä laajennoksia sekä regressio-ongelmien piirteenvalintaan soveltuvat, luokitteluun soveltuvien vastineidensa pohjalta kehitetyt, RRelief ja RReliefF [27].

Piirteidenvalinnan kääremalli (*wrapper method*) käyttää optimaalisen piirrejoukon määrittämiseksi valitun koneoppimisalgoritmin tuottamia tuloksia. Algoritmin opettaminen tehdään joko kaikilla piirrejoukon kombinaatioilla tai jonkin hakualgoritmin perusteella muodostetuilla osajoukoilla. Optimaaliseksi osajoukoksi valitaan se, joka tuottaa tarkimman tuloksen. Koska piirrejoukon paremmuuden määrittäminen tehdään suoraan opetusalgoritmia käyttäen, kääremallissa riskinä on opetuksen tuloksena muodostuvan mallin ylisovittuminen. Toisaalta optimaalinen osajoukko on tätä piirteidenvalintamenetelmää käyttäen aina sopiva käytettävälle oppimisalgoritmille. Laskennallisesti optimaalisen piirrejoukon muodostaminen kääremallilla on suodattavaa piirteidenvalintamenetelmää raskaampi vaihtoehto, mutta tyypillisesti sillä saavutetaan myös tarkempi malli. [27, s.27]

Piirteiden valinta voi myös olla integroituna koneoppimisalgoritmiin, jolloin piirteiden valinta sulautuu osaksi algoritmin oppimisvaihetta. Siinä missä suodattavaa piirteidenvalintaa sekä kääremallia voidaan tyypillisesti käyttää erilaisten koneoppimisalgoritmien yhteydessä, tällaisessa sulautetuksi piirteidenvalinnaksi kutsutussa mallissa piirteenvalinta-algoritmi sopii vain kyseisen koneoppimisalgoritmin yhteydessä käytettäväksi [28, s.137]. Guyon [28, s.162] mainitsee, että sulautetut

piirteidenvalinta-algoritmit ovat herkkiä piirteiden ylisovittumiselle opetusnäytteiden määrän ollessa pieni ja tällöin piirteidenvalintaan on monissa tapauksissa parempi käyttää suodattavaa valinta-algoritmia. Näytteiden määrän kasvaessa sulautettuun malliin perustuva piirteenvalinta kuitenkin tehostuu suodattavia algoritmeja paremmaksi.

3.4 Kernelfunktiot

Kernelfunktiot ovat symmetrisiä positiivisesti definiittejä funktioita, joiden avulla voidaan siirtää lineaarisesti separoitumattomat opetusnäytteet sellaiseen korkeampiulotteiseen piirreavaruuteen, jossa ne saadaan separoitumaan lineaarisesti [29, s.27]. Alkuperäisessä piirreavaruudessa tarkasteltuna hypertaso muodostuu tällöin epälineaariseksi. Kernelfunktioiden avulla voidaan määritellä luokittelualgoritmeille epälineaarisia päätöspintoja, mikä parantaa merkittävästi luokittelualgoritmien soveltamismahdollisuuksia käytännön luokitteluongelmiin [14, s.89]. Kernelfunktioita voidaan käyttää monissa koneoppimisalgoritmeissa, muun muassa kohdissa 3.5 ja 3.6 esiteltävissä tukivektorikoneessa ja relevanssivektorikoneessa.

Monista lineaarisista malleista – joihin tukivektorikonekin kuuluu – voidaan muodostaa duaaliesitys. Duaaliesityksessä esiintyvä sisätulo xx' voidaan korvata symmetrisellä positiivisesti definiitillä eli Mercerin ehdon täyttävällä kernelfunktiolla

$$K(\boldsymbol{x}, \boldsymbol{x'}) = \Phi(\boldsymbol{x}) \cdot \Phi(\boldsymbol{x'}), \tag{3.1}$$

jossa Φ on kuvaus d-ulottuvuuksisesta euklidisesta syöteavaruudesta $\mathbb{R}^d$ korkeampiulottuvuuksiseen avaruuteen $\mathbb{H}$:

$$\Phi : \mathbb{R}^d \mapsto \mathbb{H}. \tag{3.2}$$

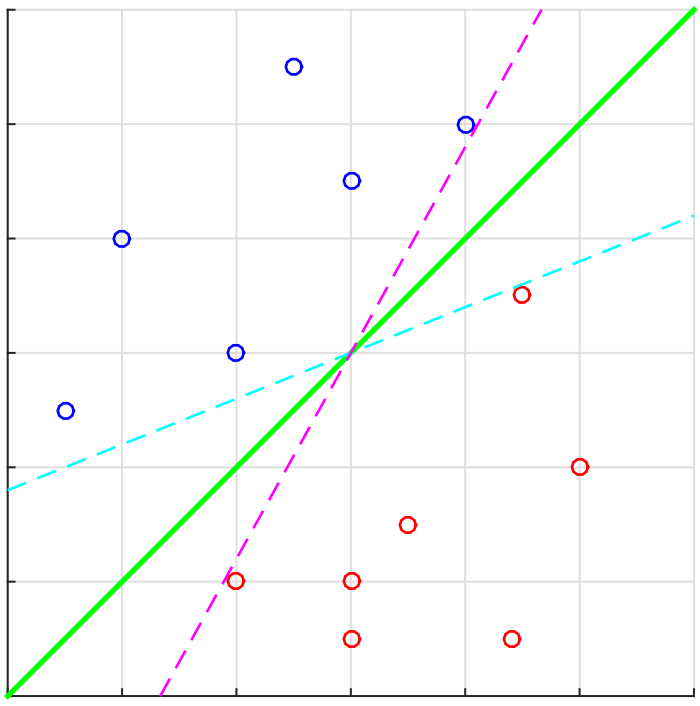
Lineaarisesti separoituvan tukivektorikoneen duaaliesityksessä (kaava 3.15) kernelfunktio korvaa siis sisätulon $x_n x_p$[14, s.90-91]. Sisätulo voidaankin tulkita lineaarisena kernelfunktiona [29, s.40]

Kernelfunktio valitaan sopivaksi ratkaistavaan koneoppimisongelmaan liittyvien näytteiden mukaan. Tukivektorikoneen kernelfunktioille määritetyt ehdot täyttäviä funktioita on monia, mutta käytännön sovelluksissa usein käytettyjä kernelfunktioita ovat Mohrin [14, s.93-94] mukaan muun muassa radiaalinen kernelfunktio $K(\boldsymbol{x}, \boldsymbol{x'}) = exp(-\frac{\|\boldsymbol{x'}-\boldsymbol{x}\|^2}{2\sigma^2})$ sekä d-asteen polynomikernel $K(\boldsymbol{x}, \boldsymbol{x'}) = (\boldsymbol{x} \cdot \boldsymbol{x'} + c)^d$.

3.5 Tukivektorikone (SVM)

Tukivektorikone (support vector machine) on ohjattuun oppimiseen perustuva luokittelijamalli, jonka esittelivät vuonna 1992 Corinna Cortes ja Vladimir Vapnik [16]. Se perustuu 1960-luvulta asti kehitettyyn tilastollisen oppimisen teoriaan, ja nykyään sitä käytetään laajalti erilaisten luokitteluun ja regressioon perustuvien koneoppimisongelmien ratkaisemiseen. Tässä työssä keskitytään ainoastaan tukivektorikoneen luokitteluominaisuuksiin.

Tukivektorikoneen perusperiaatena on muodostaa piirreavaruuteen sellainen hypertaso, joka erottaa eri luokkiin määritellyt opetusnäytteet optimaalisella marginaalilla. Marginaalilla tarkoitetaan opetusnäytteen kohtisuoraa etäisyyttä muodostetusta hypertasosta. Tukivektoreiksi valitaan sellaiset näytteet, joiden etäisyys muodostetusta hypertasosta on pienin. Nämä näytteet olisivat opetusnäytteistä vaikeimmin luokiteltavia. Kuvassa 3.1 optimaalinen hypertaso on esitetty vihreänä tasona kaksiulotteisessa piirreavaruudessa. Violetti ja vaaleansininen taso erottavat niin ikään luokat toisistaan, mutta marginaali tasoja lähimpiin näytteisiin ei ole maksimaalinen.



Kuva 3.1 *Optimaalisen hypertason määrittäminen*

SVM herätti tilastotieteellisessä yhteisössä epäilyksiä julkaisunsa aikoihin, johtuen mm. päätöksentekosääntöjen muodostamisesta korkeaulotteisen piirreavaruuden ja kernelfunktioiden perusteella. Laajamittaisten empiiristen onnistumisten siivittämänä tukivektorikoneen asema vakiintui kuitenkin standardiksi tilastollisten työkalujen joukossa. Ensimmäisiä todisteita tukivektorikoneen käyttökelpoisuudesta käytännön ongelmissa olivat hyvät tulokset AT&T Bell Labsin käsinkirjoitetun tekstintunnistustestissä ja maailmanennätys MNIST-tietokannan suorituskykytestissä.[22, s.28]

Tukivektorikone on luokittelijana binäärinen, eli sillä voidaan ratkaista sellaisia luokitteluongelmia, joihin liittyy kaksi luokkaa. Useampia luokkia käsittäviä ongelmia voidaan ratkaista tukivektorikoneita käyttäen jakamalla ongelma binäärisiksi ongelmiksi ja ketjuttamalla sitten useita binäärisiä tukivektorikoneita. Tukivektorikoneesta on olemassa myös yksiluokkainen muunnos, joka esitellään myöhemmin tässä luvussa. [15, p.338]

3.5.1 Lineaarisesti separoituvat opetusnäytteet

Tukivektorikoneen perustapauksessa opetusnäytteet separoituvat suoraan kahteen luokkaan lineaarisen hypertason erottamana, jolloin kohdassa 3.4 esiteltyjen kernelfunktioiden käyttäminen ei ole tarpeen. Esimerkki lineaarisesti separoituvasta luokitteluongelmasta on esitetty kuvassa 3.1. Tällöin lineaarisesti separoituvien opetusnäytteiden $\tau = \{\vec{\boldsymbol{x_n}}, y_n\}; n = 1, ..., m; \vec{\boldsymbol{x_n}} \in \mathbb{R}^n; y_n \in \{-1, +1\}$ luokkien -1 ja $+1$ välillä on olemassa hypertaso [14, s. 64]:

$$\boldsymbol{w}^T \cdot \boldsymbol{x} + b = 0, \tag{3.3}$$

jossa skalaari $b \in \mathbb{R}$. Hypertaso erottaa opetusnäytteet siten, että [14, s. 64]:

$$\boldsymbol{w}^T \cdot \boldsymbol{x} + b \geq 0 \; kun \; y_n = +1 \tag{3.4}$$

$$\boldsymbol{w}^T \cdot \boldsymbol{x} + b < 0 \; kun \; y_n = -1. \tag{3.5}$$

Kaikkien opetusnäytteiden pituus on N, joka määrittää myös piirreavaruuden ulottuvuuksien määrän $x_n \in \mathbb{R}^N, N \geq 1$. Luokkien väliin voidaan useissa tapauksissa sovittaa ääretön määrä edellä mainitut ehdot täyttäviä hypertasoja [14, s.64]. Tällöin optimaalisen hypertason löytämiseksi on maksimoitava luokkien välille jäävä marginaali eli luokat erottavan hypertason suuntaisten rinnakkaisten hypertasojen välinen etäisyys $\|\boldsymbol{w}\|^{-1}$, joka vastaa lausekkeen $\frac{1}{2}\|w\|^2$ minimoimista [15, s. 328].

Tästä saadaan neliöllinen optimointiongelma

$$\arg\min_{\boldsymbol{w},b} \frac{1}{2}\|w\|^2 \tag{3.6}$$

Koska datanäytteet eivät saa jäädä marginaalitasojen väliin, asetetaan seuraavat ehdot:

$$\boldsymbol{w}^T\boldsymbol{x}_n + b \geq +1 \; kun \; y_n = +1 \tag{3.7}$$

$$\boldsymbol{w}^T\boldsymbol{x}_n + b \leq -1 \; kun \; y_n = -1. \tag{3.8}$$

Ehdot 3.7 ja 3.8 voidaan esittää myös yhdellä lauseella

$$y_n(\boldsymbol{w}^T\boldsymbol{x}_n + b) \geq 1. \tag{3.9}$$

Käyttämällä Lagrangen kertoimia $a_n \geq 0, a \in [1, m]$ ja merkitsemällä symbolilla $\boldsymbol{a}$ vektoria $(a_1, ..., a_m)^T$, voidaan määritellä kaikille $\boldsymbol{w} \in \mathbb{R}^N, b \in \mathbb{R}$ ja $a \in \mathbb{R}_+^m$ Lagrangen funktio [15, s. 328]

$$L(\boldsymbol{w}, b, \boldsymbol{a}) = \frac{1}{2}\|w\|^2 - \sum_{n=1}^{m} a_n[y_n(\boldsymbol{w}^T \cdot \boldsymbol{x}_n + b) - 1] \tag{3.10}$$

$$ehdolla \; a_n \geq 0. \tag{3.11}$$

Asettamalla $L(\boldsymbol{w}, b, \boldsymbol{a})$:n osittaisderivaatat nollaksi $\boldsymbol{w}$:n ja b:n suhteen, saadaan muodostettua Karush-Kuhn-Tucker (KKT) ehdot

$$\boldsymbol{w} = \sum_{n=1}^{m} a_n y_n x_n \tag{3.12}$$

$$\sum_{n=1}^{m} a_n y_n = 0 \tag{3.13}$$

$$a_n = 0 \vee y_n(\boldsymbol{w} \cdot \boldsymbol{x} + b) = 1. \tag{3.14}$$

Kaavan 3.12 mukaan vektori $\boldsymbol{w}$ muodostuu yhdistelmästä opetusnäytteiden piirrevektoreista $x_1, ..., x_m$. Piirrevektori x_n sisällytetään tähän yhdistelmään silloin, kun $a_n \neq 0$, mikä todetaan myös kaavassa 3.14. Tällaisia vektoreita kutsutaan *tukivektoreiksi* [14, s.67].

Tukivektorit ovat opetusnäytteitä, jotka sijaitsevat piirreavaruudessa suurimman marginaalin tasoilla $y_n(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) = \pm 1$ [14, s.67]. Tällöin hypertasolle määritetty maksimaalinen marginaali voidaan sovittaa näiden vektoreiden väliin siten, että yksikään tukivektori ei tunkeudu ehdolla 3.9 määritetyn yksikkömarginaalin sisäpuolelle. Tyypillisissä luokitteluongelmissa vain osa opetusnäytteistä täyttää tu-

kivektorilta vaaditut ehdot ja tulee näin käytetyksi hypertason muodostamisessa, mikä on Bishopin [15, s.330] mukaan keskeinen tekijä tukivektorikoneen soveltuvuudelle käytännön luokittelutehtäviin. Opetettuun luokittelijamalliin käytetään usein vain pientä osaa kaikista opetusnäytteistä, mutta tähän osaan datasta sisältyy luokittelulle olennainen tieto. Tällöin tuntematonta näytettä ei tarvitse verrata kaikkiin opetusnäytteisiin sitä luokiteltaessa.

KKT-ehtoja käyttämällä eliminoidaan lausekkeesta $L(\boldsymbol{w}, b, \boldsymbol{a})$ muuttujat $\boldsymbol{w}$ ja b. Kun ratkaistaan ääriarvokohdat parametrien $\boldsymbol{w}$ ja b suhteen, saadaan duaaliesitys suurimman marginaalin ongelmasta [15, s.329], jossa maksimoidaan

$$L(a) = \sum_{n=1}^{m} a_n - \frac{1}{2} \sum_{n=1}^{m} \sum_{p=1}^{m} a_n a_p y_n y_p (x_n \cdot x_p), \tag{3.15}$$

ja jolle pätevät ehdot

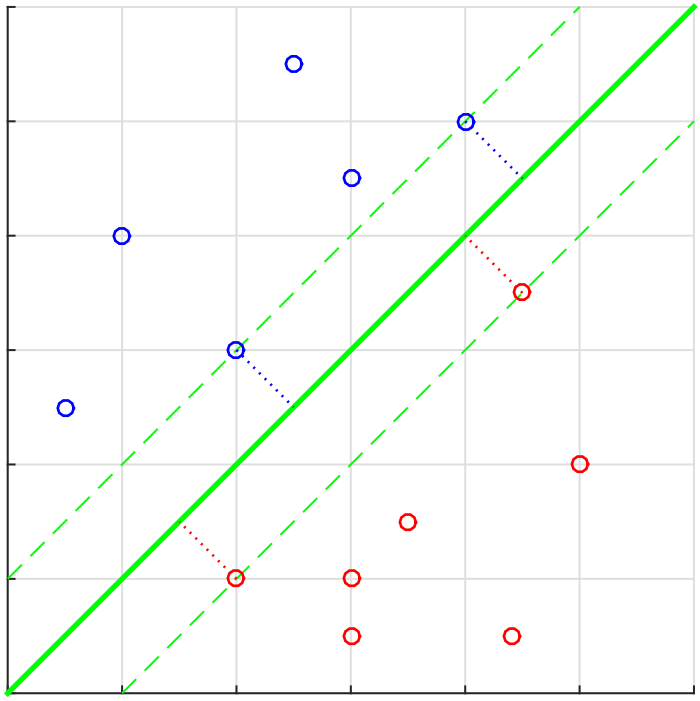
$$\begin{aligned} a_n \geq 0, n = 1, ..., m, \\ \sum_{n=1}^{N} a_n y_n = 0. \end{aligned} \tag{3.16}$$

Kvadraattisesta optimointiongelmasta 3.6 on nyt johdettu duaaliongelma. Kaikissa tapauksissa tämä ei ole kompleksisuuden näkökulmasta tarkasteltuna optimaalista, mutta se mahdollistaa kernelfunktioiden käyttämisen, jolloin tällaista luokittelijaa voidaan käyttää tehokkaasti sellaisissa piirreavaruuksissa, jossa avaruuden ulottuvuuksien määrä ylittää luokittelijalle syötettyjen näytteiden määrän.[15, s.329]

Duaaliongelma 3.15 ratkaistaan käyttämällä neliöllisen optimointiongelman ratkaisumenetelmiä, joihin ei paneuduta tässä työssä. Ratkaisun tulokseksi saadaan $\boldsymbol{a}$, josta kaavaa 3.12 käyttämällä saadaan ratkaistua $\boldsymbol{w}$. Optimaalisen hypertason määritämiseksi täytyy vielä ratkaista skalaari b. Koska tukivektorit sijaitsevat maksimaalisen marginaalin hypertasoilla, mille tahansa tukivektorille $\boldsymbol{x_n}, \boldsymbol{w} \cdot \boldsymbol{x} + b = y_n$, jolloin b ratkaistaan

$$b = y_n - \sum_{m \in S}^{m} \boldsymbol{a}_m y_m (\boldsymbol{x_m} \cdot \boldsymbol{x}), \tag{3.17}$$

jossa S kuvaa käytettävien tukivektoreiden indeksejä piirrematriisissa. Näin on saatu ratkaistua optimaalisen hypertason määrittävät $\boldsymbol{w}$ ja b. [14, p.68]



Kuva 3.2 *Optimaalinen lineaarinen hypertaso ja tukivektorit*

Kuvassa 3.2 esitetyssä luokitteluongelmaesimerkissä nähdään, kuinka opetusjoukon molemmista luokista (siniset ja punaiset vektorit) on määritetty kaksi hypertasoa lähimpänä olevaa vektoria tukivektoreiksi. Tukivektorit sijaitsevat maksimaalisen marginaalin määräämillä hypertasoilla.

3.5.2 Joustavan marginaalin tukivektorikone

Monissa käytännön luokitteluongelmissa opetusnäytteiden luokat eivät ole edellä esitetyn kaltaisesti lineaarisesti separoituvia [14, s.71]. Toisin sanoen tällaisessa tilanteessa jotkut opetusnäytteet eivät täytä tukivektorille määriteltyjä ehtoja eikä lineaarista hypertasoa pystytä täten muodostamaan.

Joustavan marginaalin (soft margin) luokittelijaksi kutsutulla tukivektorikonetoteutuksella separoitumattomuusongelma ratkaistaan määrittämällä kaikille opetusnäytteelle ns. *slack-vakio* ξ_n. Slack-vakioita käytetään optimointitehtävissä määrittämään tiukoista rajoitteista kevennettyjä versioita. Slack-vakiot mahdollistavat sen, että osa opetusnäytteistä sijoittuu luokat erottavan hypertason väärälle puolelle, eli

luokittelijan opetuksessa sallitaan luokitteluvirheitä. Tällöin opetusnäytteen määritelmä (kaava 3.9) muuttuu siten, että jokaiselle opetusnäytteille on olemassa $\xi_n \geq 0$ siten, että:

$$y_n(\boldsymbol{w}^T \boldsymbol{x}_n + b) \geq 1 - \xi_n, n = 1, ..., m. \tag{3.18}$$

Slack-vakio ξ_n kuvaa näytteen etäisyyttä siitä päätöspinnasta, jossa näytteen kuuluisi olla, jotta marginaali täyttäisi ehdon 3.9. Oikein luokittuville opetusnäytteille slack-vakion arvo on siis 0. *Poikkeaviksi havainnoiksi* tulkitaan sellaiset näytteet x_n, joille $0 < y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b) < 1$. Ne ovat luokituksensa puolesta hypertason oikealla puolella, mutta optimaalisen marginaalin sisäpuolella, jolloin niille slack-vakio $\xi_n > 0$. [14, s.71]

Joustavan marginaalin luokittelijassa tavoitteena on maksimoida luokat erottavaa hypertasoa ympäröivä marginaali siten, että slack-vakioiden summa pysyy minimaalisena. On siis minimoitava

$$C \sum_{n=1}^{m} \xi_n + \frac{1}{2}\|w\|^2, \tag{3.19}$$

jossa parametrilla $C > 0$ säädellään maksimaalisen marginaalin ja poikkeavien havaintojen aiheuttaman haitan välistä tasapainoa. Parametrin optimaalinen arvo voidaan määrittää käytännön tilanteissa esimerkiksi ristivalidoimalla. Jokaisella hypertason väärälle puolelle sijoittuvalla opetusnäytteellä $\xi_n > 1$, joten $\sum_n \xi_n$ kuvaa siten ylärajaa väärin luokiteltavien opetusnäytteiden määrälle. [15, s.332]

Kuten lineaarisesti separoituvassa tapauksessakin, myös tästä minimointitehtävästä muodostetaan Lagrangen funktio

$$L(\boldsymbol{w}, b, \boldsymbol{a}) = \frac{1}{2}\|w\|^2 + C \sum_{n=1}^{m} \xi_n - \sum_{n=1}^{m} a_n[y_n(\boldsymbol{w}^T \cdot \boldsymbol{x}_n + b) - 1 + \xi_n] - \sum_{n=1}^{m} \mu_n \xi_n \tag{3.20}$$

käyttäen Lagrangen kertoimia $a_n \geq 0$ ja $\mu_n \geq 0$. Derivoidaan muuttujien $\boldsymbol{w}$, b ja ξ_n suhteen ja asettamalla osittaisderivaatat nolliksi saadaan KKT-ehdot [14, s.73]

$$\boldsymbol{w} = \sum_{n=1}^{m} \boldsymbol{a}_n y_n \boldsymbol{x}_n \tag{3.21}$$

$$\sum_{n=1}^{m} \boldsymbol{a}_n y_n = 0 \tag{3.22}$$

$$\boldsymbol{a}_n + \mu_n = C \tag{3.23}$$

$$\boldsymbol{a}_n = 0 \vee y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b) = 1 - \xi_n \tag{3.24}$$

$$\mu_n = 0 \vee \xi = 0. \tag{3.25}$$

Optimointiongelman duaaliesitys lineaarisesti separoitumattomassa tapauksessa on identtinen lineaarisesti separoituvien näytejoukkojen duaaliesitykseen verrattaessa. Duaaliesitys on esitetty kaavassa 3.15. Duaaliesityksen ehtoihin lisätään rajoite Lagrangen kertoimelle $\mu_n \geq 0$, joka vastaa KKT-ehdon 3.23 nojalla ehtoa $a_n \leq C$. [14, s.74] Tällöin optimointiongelman ehdoiksi muodostuvat

$$\begin{aligned} 0 \leq a_n \leq C, \\ \sum_{n=1}^{m} a_n y_n = 0. \end{aligned} \tag{3.26}$$

Kuten lineaarisesti separoituvien opetusnäytteiden tapauksessa, optimointiongelma ratkaistaan käyttäen neliöllisen optimointiongelman ratkaisumenetelmiä, joita ei kuvata tässä. Ratkaistuna syntyneestä vektorista a voidaan KKT-ehtojen nojalla ratkaista skalaari b.

3.5.3 Yksiluokkainen tukivektorikone

Yksiluokkaista tukivektorikonetta opetettaessa opetusnäytteitä tarvitaan vain yhdestä luokasta. Tällöin esimerkiksi viallisia mittauskohteita tunnistettaessa opetusdataa tarvitaan pelkästään ainoastaan joko ehjistä tai rikkinäisistä mittauskohteista. Yksiluokkainen SVM soveltuu opetusdatasta poikkeavien näytteiden tunnistamiseen ja luokittelun tuloksena saadaan tieto siitä, kuuluuko luokiteltava havainto samaan luokkaan opetusdatan kanssa.

Schölkopf [30] kuvaa yksiluokkaisen tukivektorikoneen päätöspinnan muodostavan hypertason muodostamiseksi algoritmin, joka tuottaa tulokseksi +1 opetusnäytteet sisältävällä alueella piirreavaruudessa ja muilla alueilla -1. Opetusnäytteiden erottamiseksi muista alueista ratkaistaan neliöllinen ongelma

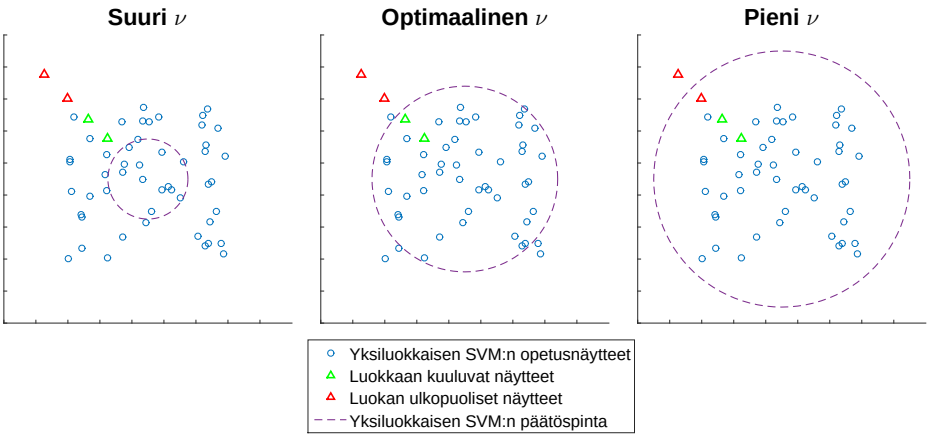
$$min_{\boldsymbol{w},\xi,\rho} \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{\nu l}\sum_{i=1}^{m}\xi_i - \rho, \tag{3.27}$$

jolle pätevät ehdot

$$(\boldsymbol{w} \cdot \Phi(\boldsymbol{x}_i)) \geq \rho - \xi_i, \xi_i \geq 0, \tag{3.28}$$

jossa $\boldsymbol{x}_1, ..., \boldsymbol{x_l}, \boldsymbol{x} \in \mathbb{R}^N$ ovat opetusnäytteet. Yksiluokkaisen SVM:n toimintaa säädellään parametrilla $\nu, 0 < \nu \leq 1$, joka kuvaa, mikä osuus tukivektoreista saa jäädä hypertason väärälle puolelle ja määrittää siten myös alarajan sille osuudelle näytteistä, jota käytetään tukivektoreina. Kuvaa 3.3 tarkastelemalla havaitaan, että

liian pienellä ν-parametrin arvolla muodostetulla yksiluokkaisella tukivektorikoneella luokkaan kuulumattomia havaintoja luokitellaan luokkaan kuuluviksi. Liian suuri arvo puolestaan aiheuttaa sen, että luokkaan kuuluvia havaintoja luokitellaan luokkaan kuulumattomiksi. Optimaalinen arvo parametrille ν voidaan etsiä esimerkiksi ristivalidointia käyttäen.



Kuva 3.3 *ν-parametrin vaikutus yksiluokkaisen tukivektorikoneen päätöspintaan.*

Tukivektoriden määrittämiseksi muodostetaan Lagrangen funktio

$$L(\boldsymbol{w},\xi,\rho,\alpha,\beta) = \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{\nu l}\sum_{i=1}^{m}\xi_i - \rho - \sum_{i=1}^{m}\alpha_i((\boldsymbol{w}\cdot\Phi(\boldsymbol{x}_i)) - \rho + \xi_i) - \sum_{i=1}^{m}\beta_i\xi_i \quad (3.29)$$

käyttäen kertoimia $\alpha_i, \beta_i \geq 0$. Asettamalla osittaisderivaatat muuttujien $\boldsymbol{w}, \xi$ ja ρ suhteen nolliksi, saadaan

$$\boldsymbol{w} = \sum_{i=1}^{m}\alpha_i\Phi(\boldsymbol{x}_i), \quad (3.30)$$

$$\alpha_i = \frac{1}{\nu l} - \beta_i \leq \frac{1}{\nu l}, \quad (3.31)$$

$$\sum_{i=1}^{m}\alpha_i = 1. \quad (3.32)$$

Tukivektoreiksi valitaan sellaiset opetusjoukon vektorit joukossa, joille $\alpha_i > 0$ [30, s.4]. Ongelman duaaliesitys muodostetaan sijoittamalla edellä määritetyt osittaisderivaatat Lagrangen funktioon 3.29 ja ottamalla käyttöön kernelfunktiot määritelmän 3.1 mukaisesti:

$$\min_{\alpha}\frac{1}{2}\sum_{ij}^{m}\alpha_i\alpha_j k(\boldsymbol{x}_i,\boldsymbol{x}_j), \quad (3.33)$$

jolle pätevät ehdot

$$\begin{aligned} 0 \leq \alpha_i \leq \frac{1}{\nu l}, \\ \sum_i^m \alpha_i = 1. \end{aligned} \tag{3.34}$$

Koska tukivektoreille pätee $\alpha_i, \beta_i > 0$, kvadraattisen ongelman rajoitteiden 3.28 epäyhtälöt muuttuvat yhtälöiksi. Tätä tietoa hyödyntäen voidaan vakio ρ ratkaista tukivektoreille kaavasta

$$\rho = (\boldsymbol{w} \cdot \Phi(\boldsymbol{x}_i)) = \sum_j^m \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j). \tag{3.35}$$

Yksiluokkaisessa tukivektorikoneessa käytettävät kernelfunktiot $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ voivat olla samoja kuin binäärisessä tukivektorikoneessa. Schölkopf [30, s.11] mainitsee, että usein parhaat tulokset saadaan käyttämällä kernelfunktiota, joka sovittaa hypertason piirreavaruuteen sijoittuneiden piirteiden keskikohdan perusteella. Tällaisella kernelfunktiolla muodostettu hypertaso rajaa luokkaan kuuluvat näytteet täysin luokkaan kuulumattomista näytteistä.

3.6 Relevanssivektorikone (RVM)

Relevanssivektorikone eli RVM (relevance vector machine) on bayesilaiseen päättelyyn perustuva regressioon ja luokitteluun soveltuva koneoppimisalgoritmi, jonka Michael Tipping esitteli vuonna 2000 [17]. Tässä työssä keskitytään ainoastaan relevanssivektorikoneen luokitteluominaisuuksiin. RVM tuottaa tulokseksi todennäköisyyden jonkin väitteen paikkaansapitävyydelle joidenkin havaintojen perusteella. Väite voi olla esimerkiksi *"mitattu kohde on viallinen"* ja havaintona on kohteelle suoritettu mittaus, jolloin RVM tuottaa tuloksena todennäköisyyden, jolla mitattu kohde on viallinen. Mikäli todennäköisyysarvosta tarvitaan binäärinen luokittelupäätös, se muodostetaan RVM:n tuottaman todennäköisyyden ja ennalta määritetyn tai jollain päätöksentekoalgoritmilla lasketun luokkien välisen raja-arvon (threshold) perusteella. [17]

Relevanssivektorikone on erikoistapaus harvasta lineaarisesta mallista, jossa kantafunktiot muodostetaan opetusnäytteille kernelfunktiota Φ käyttämällä:

$$y(\boldsymbol{x}) = \sum_{i=1}^m w_i \Phi(x, x_i). \tag{3.36}$$

Vaikka relevanssivektorikone perustuu tukivektorikoneen tavoin lineaariseen mal-

liin, kernelfunktion ei relevanssivektorikoneen tapauksessa tarvitse täyttää Mercerin ehtoa, joka määrittää, että kernelfunktion Φ on oltava jatkuva ja positiivisesti definiitti. Tämä mahdollistaa useiden kernelfunktioiden käyttämisen, jolloin lineaarinen malli on muotoa

$$y(\boldsymbol{x}) = \sum_{m=1}^{M} \sum_{i=1}^{m} w_{mi} \Phi_m(x, x_i), \tag{3.37}$$

jossa Φ_m kuvaa kernelfunktioita. RVM:n harvuus mahdollistaa sopivimman kernelin valitsemisen jokaisessa pisteessä karsimalla epäolennaiset kernelfunktiot. On kuitenkin mahdollista, että yhteen pisteeseen vaikuttaa kaksi kernelfunktiota. [31]

Relevanssivektorikoneen käyttämiseksi luokitteluun muodostetaan posterioriset todennäköisyydet näytteen $\boldsymbol{x}$ kuulumiselle eri luokkiin. Muodostetaan yleistetty lineaarinen malli sijoittamalla $y(\boldsymbol{x})$ logistiseen funktioon $\sigma(y) = 1/(1 + e^{-y})$ ja ilmaisemalla todennäköisyys

$$P(\boldsymbol{t}|\boldsymbol{w}) = \prod_{n=1}^{m} \sigma\{y(\boldsymbol{x}_n)\}^{t_n} [1 - \sigma\{y(\boldsymbol{x}_n)\}]^{1-t_n}, \tag{3.38}$$

jossa t kuvaa luokkatunnisteita, $\boldsymbol{w}$ on joukko painokertoimia ja $\{x_n\}_{n=1}^{N}$ on opetusnäytteiden joukko. [17]

Toisin kuin tukivektorikoneen tapauksessa, jossa tavoitteena on tuottaa suurimman marginaalin luokittelija, relevanssivektorikoneessa muodostetaan mahdollisimman harva malli. Tämän saavuttamiseksi painokertoimet $\boldsymbol{w}$ muodostetaan Bayesialaisen päättelyn keinoin ja jokaiselle painokertoimelle $\boldsymbol{w}_n$ muodostetaan siihen liittyvä hyperparametri α_n. Hyperparametrit optimoidaan siten, että luokittelun kannalta epäolennaiset vektorit saavat arvokseen nollaa lähestyvän arvon. Tuntemattomia näytteitä luokiteltaessa nollaa lähestyviä painokertoimia ja niihin liittyviä kernelfunktioita ei huomioida tuntemattomia, jolloin tuloksena on harva malli.[17]

3.7 RVM:n ja SVM:n vertailu

Relevanssivektorikone on Bayesialaiseen päättelyyn perustuva koneoppimismenetelmä, joka poistaa tiettyjä tukivektorikoneeseen liityviä rajoitteita, ja jonka toimintaperiaatteita on tehostettu tukivektorikoneeseen verrattuna. Yhtenä tärkeimmistä eroavaisuuksista Tipping [17] mainitsee relevanssivektorikoneen kyvyn tuottaa ulostulona todennäköisyysarvoja. Siinä missä tukivektorikone kykenee tuottamaan vain binäärisen luokittelutuloksen, relevanssivektorikoneen tuottamat posterioriset todennäköisyydet ilmaisevat myös ennusteen varmuutta. Myös tukivektorikonetta varten on olemassa joitain tapoja, joilla posteriorisia todennäköisyyksiä voidaan

muodostaa, mutta Tippingin mukaan niitä ei voida pitää luotettavina.

Relevanssivektorikone ei ole myöskään rajoitettu vain binääriseen luokitteluun, vaan sitä voidaan käyttää useampiluokkaisten luokitteluongelmien ratkaisemiseen ilman useampien luokittelijoiden kouluttamista ja ketjuttamista [17]. Kuten kohdassa 3.5 todettiin, useampiluokkaisten ongelmien ratkaisuun tukivektorikoneita voidaan käyttää vain useampia luokittelijoita kouluttamalla ja yhdistelemällä niiden tuottamia tuloksia.

Tukivektorikone on harva malli, eli mallin muodostamiseksi käytetään vain osaa opetusnäytteistä. Tästä huolimatta mallin muodostamiseen käytettyjen tukivektoreiden määrä kasvaa lineaarisesti opetusjoukon kasvaessa, mikä puolestaan kasvattaa mallin laskennallista monimutkaisuutta. Relevanssivektorien määrä puolestaan pysyy huomattavasti alhaisempana, sillä opetusnäytteistä valitaan mallissa käytettäväksi vain kaikista relevanteimmat automaattista relevanssinmääritystä (automatic relevance determination) hyödyntäen.[17]

Tukivektorikonetta opetettaessa joudutaan usein etsimään optimaalinen arvo parametrille C, jolla säädellään maksimaalisen marginaalin ja poikkeavien havaintojen aiheuttaman haitan välistä tasapainoa. Käytännössä parametrin etsiminen voidaan tehdä ristivalidoinnilla. Relevanssivektorikoneessa tätä parametria ei ole, mikä tosin aiheuttaa sen, että ratkaistava optimointiongelma on usein epälineaarinen.[31] Luokittelijan opetusaika saattaa siksi olla tukivektorikonetta pidempi, mutta tuntemattomia näytteitä luokiteltaessa relevanssivektorikone on tyypillisesti huomattavasti tukivektorikonetta nopeampi.[15, s.356]

Tukivektorikoneessa tukivektoreiksi tulee valituksi tyypillisesti sellaiset opetusnäytteet, jotka sijaitsevat suurimman marginaalin rajalla. Tällaiset näytteet ovat toisin sanoen opetusjoukon vaikeimmin luokiteltavia. Relevanssivektorikoneen päätöspinta puolestaan muodostetaan relevanteimmiksi havaittujen näytteiden perusteella, jotka eivät välttämättä ole päätöspintaa lähimpänä olevia näytteitä.[31]

Tukivektorikoneessa käytettävien kernelfunktioiden on täytettävä Mercerin ehdot, eli niiden tulee olla symmetrisiä ja positiivisesti definiittejä. Relevanssivektorikoneessa tätä rajoitetta ei puolestaan ole.[17]

4. VAURIONTUNNISTUSKOMPONENTTI

Vauriontunnistuskomponentin avulla tunnistetaan kappaleiden kunto niistä mitattujen värähtelymittausten perusteella. Vaurioita tunnistetaan ohjattuun oppimiseen perustuvien luokittelualgoritmien avulla. Vauriontunnistuskomponentissa käytettävien algoritmien valinta kuvattiin kohdassa 3.2.

Tässä luvussa kuvataan komponentin suunnittelun lähtökohdat ja sille asetetut vaatimukset. Vaatimuksien perusteella suunnitellaan ja kuvataan komponentin arkkitehtuuri. Lisäksi valitaan komponentin luokittelutehtäviin soveltuvat tukivektorikone- ja relevanssivektorikonetoteutukset ja kuvataan vauriontunnistuskomponentin eri osien toiminta pääpiirteittäin.

4.1 Suunnittelun lähtökohdat

Vauriontunnistuskomponentin kehittämisen taustalla on asiakasyrityksen tarve kyetä havaitsemaan vaurioita kappaleista värähtelymittausten perusteella. Komponentin syötteenä käytettävät näytteet tuotetaan asiakkaan kehittämillä värähtelymittalaitteilla, jotka on suunniteltu erityisesti asiakasyritystä kiinnostavien kappaleiden mittaamista varten. Komponentin suunnittelussa vältetään liiallista sitoutumista tiettyyn mittalaitteeseen, sillä asiakkaan mittalaitekehitys on edelleen aktiivista ja tulevaisuudessa komponenttia tullaan käyttämään useilla erilaisilla mittalaitteilla mitatuilla näytteillä. Tavoitteena on kehittää yleiskäyttöinen työkalu, jota voidaan käyttää värähtelymittauksien analysointiin käytetystä mittalaitteistosta riippumatta.

Vauriontunnistuskomponentin tulee tässä vaiheessa kehitystä kyetä ilmaisemaan, onko mitattu kappale ehjä vai vaurioitunut. Kun tällainen binäärinen luokittelu on todettu toimivaksi, voidaan vaurioanalyysiä tulevaisuudessa kehittää hienojakoisemmaksi esimerkiksi ilmaisemalla vaurioiden sijainti kappaleessa, arvioimalla kappaleen jäljellä olevaa käyttöikää tai tunnistamalla kohdassa 2.3 mainittuja vaurioiden tyyppejä.

Kappaleen eheyden tai vaurioituneisuuden määrittelyn tulee perustua aikaisemmin

mitattuihin näytteisiin, joissa mittauskohteen kunto tunnetaan. Kunnoltaan tuntemattomien näytteiden luokittelu tehdään tunnettuihin näytteisiin perustuen. Kohdassa 3.2 esitetyn koneoppimisalgoritmien vertailun perusteella tällaiseen luokittelutehtävään hyvin soveltuviksi algoritmeiksi havaittiin tukivektorikone ja relevanssivektorikone. Luokittelijaa opetettaessa määritetään, mitä luokittelualgoritmia käyttäen tuntemattomien näytteiden luokitteluun käytettävä luokittelijamalli muodostetaan. Sopiva luokittelualgoritmi valitaan luokittelutehtävän mukaan, ja kerrallaan käytössä voi olla ainoastaan yksi algoritmi.

Tukivektorikoneesta ja relevanssivektorikoneesta on olemassa avoimeen lähdekoodiin perustuvia toteutuksia, joita vauriontunnistuskomponentissa voidaan hyödyntää. Kolmannen osapuolen toteuttamia komponentteja käytetään, sillä verraten vaativien luokittelualgoritmien toteuttaminen itse on sekä virhealtista että aikaa vievää ja tulee näin ollen kalliiksi myös asiakkaalle.

Komponenttia tullaan käyttämään osana pilvipalveluympäristöä, jossa asiakkaan kehittämä isäntäohjelmisto toimii. Isäntäohjelmisto on toteutettu Microsoftin .NET -sovelluskehystä käyttäen, joten vauriontunnistuskomponentin tulee tarjota isäntäjärjestelmälle .NET-yhteensopiva rajapinta.

Näytteiden käsittelyyn ja luokittelualgoritmiin liittyvä laskenta toteutetaan MATLAB-ympäristössä. MATLAB tarjoaa kattavat työkalut esikäsittelyssä tarvittavien suodinten suunnitteluun ja piirteiden laskennassa käytettäviin vektori- ja matriisioperaatioihin. Lisäksi useiden luokittelualgoritmien toteutuksia on saatavilla MATLAB-yhteensopivina ja aikaisemmin asiakkaalle tehdyissä projekteissa on toteutettu tässä työssä uudelleenkäytettäväksi soveltuvia MATLAB-laskentafunktioita.

4.2 Luokittelualgoritmien toteutukset

Vauriontunnistuskomponentissa käytettävien luokittelualgoritmien toteutuksina käytetään kolmansien osapuolten tarjoamia ratkaisuja. Valmiskomponenttien käyttäminen on perusteltua, sillä tällaisten monimutkaisten algoritmien toteuttaminen alusta asti on virhealtista ja aikaavievää. Seuraavassa esitellään joitain tukivektorikone- ja relevanssivektorikonetoteutuksia ja valitaan niistä parhaiten soveltuvat toteutukset käytettäviksi vauriontunnistuskomponentissa.

4.2.1 LibSVM

LibSVM [32] on avoimeen lähdekoodiin perustuva tukivektorikonekirjasto, jota on kehitetty Taiwanin kansallisyliopistossa vuodesta 2000 lähtien ja kirjaston kehitys-

työ on edelleen aktiivista. Kirjastosta on olemassa toteutukset C++:lla, Javalla ja Pythonilla. Monissa muissa ympäristöissä sitä voidaan käyttää erilaisten kääreiden ja rajapintojen avulla. Vauriontunnistuskomponentissa käytettävän MATLABin lisäksi LibSVM:ää voidaan käyttää monissa muissakin ohjelmointiympäristöissä, mm. .NET, Python ja Haskell. LibSVM on julkaistu yksinkertaistetun BSD-lisenssin (BSD 3-Clause license) [33] alaisuudessa, jonka mukaan sen käyttäminen kaupallisissa projekteissa on sallittua ilman lähdekoodin julkaisemista.

SVM-toteutuksen lisäksi LibSVM sisältää myös työkaluja mm. luokittelijalle syötettävän datan skaalaukseen, merkitsevien piirteiden löytämiseen ja optimaalisten luokitteluparametrien määrittämiseen. Kirjaston dokumentaatio on kattavaa ja pelkän käyttöopastuksen lisäksi saatavilla on myös tieteellisiä julkaisuja kirjaston tarjoamien algoritmitoteutusten toiminnasta. Binäärisen tukivektorikoneen lisäksi LibSVM sisältää toteutuksen yksiluokkaiselle tukivektorikoneelle sekä pystyy myös ratkaisemaan useamman kuin kahden luokan luokitteluongelmia.

4.2.2 MATLAB Statistics and Machine Learning Toolbox

Mathworks tarjoaa MATLABin ominaisuuksien laajentamiseksi useita työkalukokoelmia, joista Statistics and Machine Learning Toolbox (SMLT) [34] sisältää koneoppimiseen liittyviä työkaluja sekä algoritmien toteutuksia. Kokoelma sisältää esimerkiksi tukivektorikonetoteutuksen sekä työkaluja mm. piirteiden valintaan, ristivalidointiin sekä luokittelutarkkuuden vertailuun.

SMLT, kuten muutkin Mathworksin tarjoamat työkalukokoelmat, on kaupallinen tuote ja hinnoittelultaan verraten kallis kaupallisessa käytössä. Lisenssin ostamisen jälkeen työkalukokoelman algoritmien käyttäminen loppukäyttäjälle jaeltavissa sovelluksissa on sallittua ilman lähdekoodien julkaisemista. Työkalujen dokumentaatio on kattavaa ja osa koneoppimistyökaluista integroituu myös MATLABin käyttöliittymään. Mathworks tarjoaa työkalukokoelman ostajalle myös käyttöopastusta mm. ilmaisten videoseminaarien muodossa.

4.2.3 Pattern recognition toolbox (PRT)

Pattern recognition toolbox (PRT) [1] on avoimeen lähdekoodiin perustuva työkalukokoelma, jossa yhtenäisen rajapinnan taakse on paketoituna useita hahmontunnistusalgoritmeja. Se on toteutettu MATLABilla, mutta käyttää algoritmitoteutuksina joitain C-kielisiä ulkoisia kirjastoja. Kirjaston käyttäminen tapahtuu kuitenkin täysin MATLAB-rajapinnan kautta.

PRT tarjoaa työkalut mm. luokitteluun, hahmontunnistusalgoritmien ketjuttamiseen, klusterointiin ja hahmontunnistusdatan visualisoimiseen. PRT on julkaistu MIT-lisenssin [35] alaisuudessa, joten sitä voidaan käyttää myös kaupallisissa projekteissa ilman lähdekoodin julkaisemista.

Työkalukokoelma sisältää muiden koneoppimisalgoritmien ohella myös relevanssivektorikoneen toteutuksen. Lisäksi PRT tarjoaa erilaisia päätöksentekoalgoritmeja RVM:n tuottamien todennäköisyysarvojen muuttamiseksi binäärisiksi luokittelutuloksiksi.

4.2.4 SparseBayes

SparseBayes on relevanssivektorikoneen kehittäjän Michael Tippingin julkaisema MATLAB-työkalukirjasto, jonka versio 1.1 sisältää relevanssivektorikoneen toteutuksen. Kirjaston uusimmassa 2.0-versiossa relevanssivektorikoneen toteutusta ei enää ole, sillä uuden version tavoitteena on tarjota yleiskäyttöiset työkalut Sparse Bayesian -mallinnukseen ja jättää huomioitta erikoistapaukset, joihin relevanssivektorikonekin lukeutuu [36]. Varsinainen relevanssivektorikonetoteutus jää siis kirjaston käyttäjän toteutettavaksi.

Kirjaston 1.1-versio tarjoaa perustyökalut relevanssivektorikoneen käyttämiseksi sekä regressio- että luokittelukäytössä. Kehittäjä kuitenkin mainitsee kirjaston mukana toimitettavassa dokumentaatiossa [37], että kirjasto on tarkoitettu perustaksi käytettävämmän ja vakaamman relevanssivektorikonekirjaston kehittämiseen. Kirjasto on lisensoitu GPL-lisenssin [38] alaisuudessa, joten sen käyttäminen velvoittaa ohjelman lähdekoodien julkaisemiseen.

4.2.5 Dlib

Dlib [39] on avoimeen lähdekoodiin perustuva kirjasto, joka sisältää useita koneoppimisalgoritmien toteutuksia sekä valikoiman koneoppimistyökaluja. Kirjastoon sisältyy paljon myös täysin koneoppimiseen liittymättömiä työkaluja. Dlib on kattavasti dokumentoitu ja myös kirjaston käyttöesimerkkejä tarjotaan runsaasti. Muiden koneoppimisalgoritmien ohella se sisältää toteutukset myös tukivektori- relevanssivektorikoneluokittelijoista. Dlib on julkaistu Boost-lisenssin alaisuudessa, joten sen käyttö sallitaan myös kaupallisissa projekteissa ohjelman lähdekoodeja julkaisematta.

Dlib on toteutettu C++-kielellä, mutta sen integroiminen MATLAB-komponenttiin on mahdollista MATLABin tarjoamien työkalujen avulla. C++-kirjaston tarjoamien

metodien kutsuminen ja paluuarvojen käsittely on kuitenkin monivaiheisempaa ja siten myös virhealttiimpaa verrattuna natiivien MATLAB-funktioiden käyttämiseen.

4.2.6 Algoritmitoteutuksien valinta

Vauriontunnistuskomponenttiin tarvitaan toteutukset SVM- ja RVM-luokittelijaalgoritmeille sekä yksiluokkaiselle tukivektorikoneelle. Luokittelijalle syötettävät piirteet lasketaan MATLABissa, joten luokittelijatoteutusta tulee voida käyttää MATLAB-komponentista. Vaikka C++-kirjastoja voidaan käyttää MATLABista, niiden kutsuminen, tulosten käsittely sekä virheenkäsittely on kuitenkin natiivien MATLAB-funktioiden käyttämistä monimutkaisempaa. Ensisijaisesti pyritään siis valitsemaan käytettäväksi natiiveja MATLAB-toteutuksia, jolloin dlib rajataan mahdollisista toteutusvaihtoehdoista pois.

Myös LibSVM on C-kirjasto, mutta sen ohessa tarjotaan myös kirjaston kehittäjien toteuttama MATLAB-kääre, joka piilottaa kommunikoinnin C-kirjaston metodien kanssa. Kirjastoa voidaan siis käyttää MATLABin tietotyyppejä ja tavallisia funktiokutsuja hyödyntäen.

Vauriontunnistuskomponentin lähdekoodia ei voida julkaista avoimesti, joten käytettävän toteutuksen lisenssin tulee olla sellainen, että se ei velvoita lähdekoodin julkaisuun. Tähän soveltuvia lisenssejä ovat esimerkiksi Boost, MIT ja BSD -lisenssit. Esitellyistä luokittelijavaihtoehdoista ainoastaan SparseBayes RVM-toteutus on lähdekoodien julkaisuun velvoittavan GPL-lisenssin alaisuudessa.

Vauriontunnistuskomponentin relevanssivektorikonetoteutuksena käytetään siis Pattern recognition toolboxin RVM-toteutusta, sillä se on ainoa vertailuun sisällytetty ja edellä kuvatut vaatimukset täyttävä relevanssivektorikonetoteutus. Tukivektorikonetoteutuksena käytetään LibSVM-kirjastoa. MathWorksin Statistics and Machine Learning Toolbox -työkalukirjastoon verrattuna LibSVM on ilmainen ja sisältää lisäksi myös yksiluokkaisen tukivektorikoneen toteutuksen, jolloin kirjastoa käyttämällä sekä binäärinen että yksiluokkainen tukivektorikone voidaan opettaa yhtenäistä rajapintaa hyödyntäen.

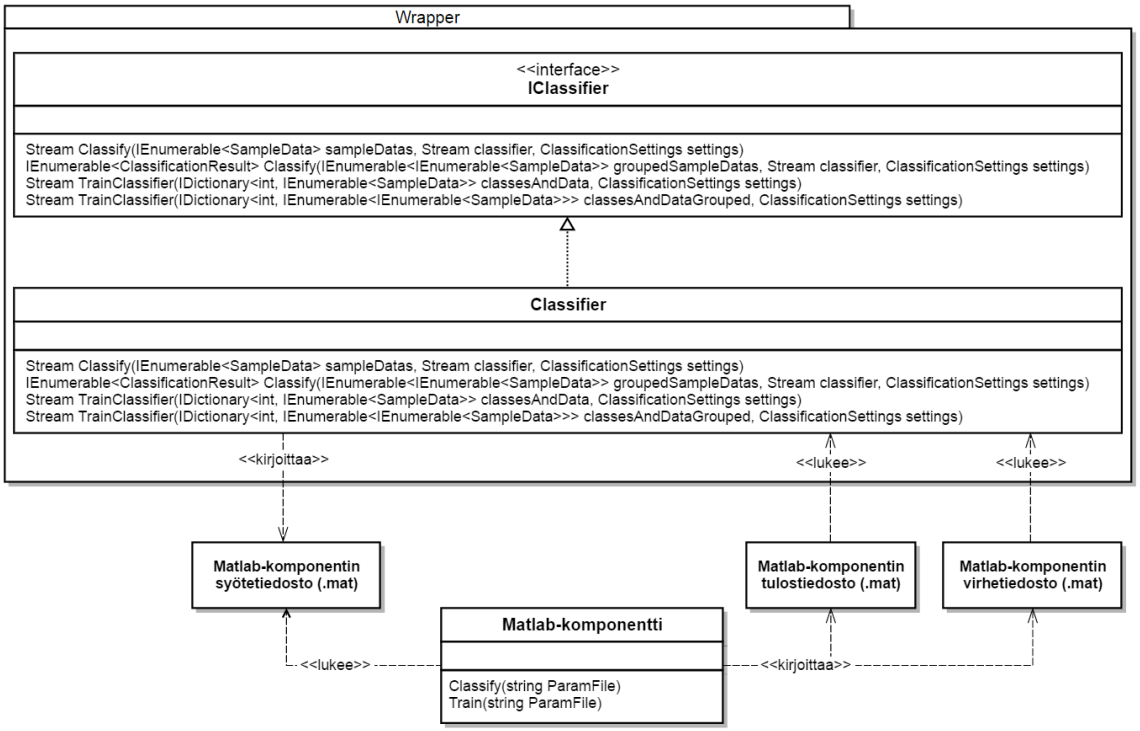
LibSVM tarjoaa MathWorksin ratkaisua kattavammat mahdollisuudet tukivektorikoneen konfiguroimiseksi. SMLT:n yksinkertaisempi lähestymistapa voi olla joissain sovelluksissa hyvä ja nopea tapa saada tukivektorikone käyttöön, mutta SVM-algoritmin sisäisen toiminnan tarkkailemiseksi ja sen hienosäätämiseen LibSVM tarjoaa paremmat työkalut. Vauriontunnistuskomponentin kehityksessä luokittelijaalgoritmin toiminnan syvempi tarkkailu ja säätäminen voi osoittautua hyödylliseksi.

LibSVM tukee myös moniluokkaisia tukivektorikonetta, jota saatetaan vauriontunnistuskomponentin jatkokehityksessä tulla hyödyntämään.

4.3 Arkkitehtuuri

Vauriontunnistuskomponentti integroituu Microsoft .NET-pohjaiseen isäntäjärjestelmään, mutta laskenta suoritetaan MATLAB-ympäristössä. Tarvitaan siis keino kutsua MATLABilla tuotettua laskentakomponenttia NET-isäntäjärjestelmän kautta ja siirtää tietoa näiden välillä. MATLABiin on saatavilla Compiler -lisäosa, jolla MATLAB-ympäristössä tuotettu ohjelmakoodi voidaan kääntää komentoriviltä suoritettavaksi ohjelmaksi (.exe). Käännetyn ohjelman käyttämiseksi ei tarvita MATLAB-asennusta lisensseineen, vaan ohjelmien ajamiseen riittää vapaasti jaeltava MATLAB Runtime -kirjasto [40].

Isäntäjärjestelmään integroitumista varten määritellään yksinkertainen .NET-rajapinta, joka tarjoaa metodit luokittelijamallin muodostamista sekä tuntemattomien näytteiden luokittelemista varten. Rajapinnan tarjoamat metodit on esitetty kuvassa 4.1. Rajapinnassa määritellään myös luokittelijan käyttöön tarvittavat tietotyypit. Lisäksi toteutetaan rajapinnan toteuttava kääre (wrapper). Kääremallin myötä isäntäjärjestelmästä ei kutsuta suoraan komentorivipohjaista laskentakomponenttia, vaan MATLAB-komponentti on piilotettu kääreeseen, joka sisältää toiminnot MATLAB-komponentin kutsumiseen, virheiden käsittelemiseen sekä tiedon siirtämiseen .NET- ja MATLAB-ympäristöjen välillä.



Kuva 4.1 *Vauriontunnistuskomponentin kääre-arkkitehtuuri ja komponenttien välinen kommunikointi.*

Kuvassa 4.1 on esitetty myös kääreen ja MATLAB-komponentin välinen viestintä. Viestintä tapahtuu levylle kirjoitettavia MATLABin MAT-tiedostomuodossa olevia asetustiedostoja käyttäen. Kääre kirjoittaa rajapinnalle argumenttina annetut luokitteluparametrit MAT-tiedostoon, jonka jälkeen MATLAB-komponenttia kutsutaan. MATLAB-komponentille parametrina annetaan kyseinen MAT-tiedosto, jonka se lukee ja käsittelee. Paluuarvot MATLAB-komponentilta kääreen suuntaan siirretään niin ikään MAT-tiedostojen välityksellä. Kääre tarkkailee, milloin MATLAB-komponentti lopettaa toimintansa ja lukee sen jälkeen ennalta määritellystä sijainnista MATLAB-komponentin levylle kirjoittaman tiedoston, joka voi sisältää luokittelijamallin tai luokittelun tuloksen. Suorituksen keskeytyessä kirjoitetaan levylle lisäksi virheilmoitustiedosto, josta ilmenee MATLAB-komponentin suorituksen keskeytymisen syy.

4.4 Virheenkäsittely

Virheenkäsittelyä on toteutettu sekä kääreeseen että MATLAB-komponenttiin. Kääreessä pyritään havaitsemaan ja käsittelemään mahdollisimman suuri osa virheistä jo ennen kuin MATLAB-komponenttia kutsutaan. MATLAB-komponentin kutsuminen on hidasta, sillä MATLABin ajonaikainen kirjasto joudutaan lataamaan joka

kutsukerran yhteydessä uudelleen. Kaikki tunnetut virheet käsitellään keskeyttämällä ohjelman suoritus hallitusti ja heittämällä poikkeus.

Kääre tarkistaa ennen MATLAB-komponentin kutsumista, että kaikki mittadatatiedostot ja luokittelijan asetuksissa määritellyt kanavat ovat saatavilla. Lisäksi tarkistetaan, ovatko käytettävät kernelfunktiot valittuun luokittelijaan liittyviä ja onko valittuja kernelfunktioita oikea määrä suhteessa käytettävään luokittelijaan. Luokkien tunnisteet on määritelty niin, että niiden arvoina voi olla ainoastaan 0 tai 1. Jos havaitaan tuntematon luokkatunniste, heitetään poikkeus.

MATLAB-komponentissa tapahtuvat poikkeukset keskeyttävät komponentin toiminnan ja virheviesti kirjoitetaan silloin levylle erilliseen tiedostoon. Virheviestit kirjoitetaan erilleen luokittelijamallin tai luokittelun tulokset sisältävästä tiedostosta, koska kääreessä MAT-tiedostojen käsittelyyn käytetty CSMatIO-kirjasto ei pysty tulkitsemaan sellaisia MAT-tiedostoja, jotka sisältävät MATLABin standardityypeistä poikkeavia olioita. PRT tallentaa RVM-luokittelijamallin standardityypeistä poikkeavana *prtClassRvm*-oliona, jolloin mitään tulostiedoston kenttiä ei pystytä tulkitsemaan CSMatIO:ta käyttäen.

MATLAB-komponentin toiminnan loppuessa kääre tarkistaa, onko virheilmoituksen sisältävää tiedostoa olemassa. Tiedoston löytyessä heitetään virheen sisältöä vastaava poikkeus, joka ohjataan isäntäjärjestelmälle. Virhetilanteen käsittely tästä eteenpäin on isäntäjärjestelmän vastuulla.

4.5 Syötetiedostot

Vauriontunnistuskomponentille syötettävät mittadatatiedostot ovat kiihtyvyysanturin tuottamia WAV-muotoisia aaltoääniä. Tällä hetkellä käytössä olevan mittalaitteiston näytteenottotaajuus on 5KHz, jolloin korkein havaittava taajuus eli ns. Nyquistin taajuus [41, s.4] on 2500Hz. Jokaiseen mittaukseen liittyy aina vähintään kaksi aaltoäänitiedostoa: heräte- ja vastesignaali.

Herätesignaali mitataan herätevasarassa olevasta voima-anturista tai joissain mittalaitteissa vasaran iskukohdan lähelle sijoitetusta kiihtyvyysanturista. Vastesignaali puolestaan mitataan mittalaitteen kiihtyvyysantureilla, joita voi mittalaitteesta riippuen olla yksi tai useita. Mittauskertojen välisen hajonnan pienentämiseksi vauriontunnistuskomponentti tukee myös syötedatan koostamista useista mittauksista mediaania tai keskiarvoa käyttäen.

4.6 .NET-rajapinta ja wrapper

IClassifier on luokittelijakomponentille määritelty .NET -rajapinta, joka tarjoaa tarvittavat metodit luokittelijan opettamiseksi sekä näytteiden luokittelemiseksi. Lisäksi siinä määritellään luokittelijan toimintaan liittyviä tietotyyppejä. Kuvassa 4.1 on kuvattu rajapinnan tarjoamat metodit ja kommunikointi MATLAB-komponentin kanssa.

Rajapinnassa määritellyistä tietotyypeistä komponentin käyttämisen kannalta oleellisimpia ovat *SampleData*, *ClassificationSettings* ja *ClassificationResult*. *SampleData* sisältää yksittäisen näytteen tiedot. Näytteeseen liittyy aina uniikki tunniste sekä hakemisto, joka sisältää mittalaitteen jokaisen kanavan tuottaman näytteen. Kanavia on oltava aina vähintään kaksi: heräte ja vaste. Kanavien määrälle ei ole määritetty ylärajaa.

ClassificationSettings-luokka pitää sisällään kaikki luokitteluasetukset. Se sisältää kentät, joiden avulla voidaan luokittelijaa opetettaessa ja luokiteltaessa määrittää mm. analysoitavat taajuusalueet, niistä laskettavat piirteet, näytteille tehtävät esikäsittelyt, käytettävä luokittelija-algoritmi sekä luokittelija-algoritmissa käytettävät kernelfunktiot. Lisäksi se sisältää kenttiä, joilla luokittelijoiden toteutuksille voidaan ohjata mahdollisia lisäargumentteja. *ClassificationResult*-luokan avulla kuvataan yksittäisen näytteen luokittelun tulokset. Se sisältää aina vähintään luokittelualgoritmin määrittämän luokan. Mikäli luokittelija-algoritmina käytetään relevanssivektorikonetta, luokittelijan tuloksiin liitetään myös todennäköisyydet näytteen kuulumiselle kuhunkin luokkaan.

Rajapinnan toteuttavan *Classifier*-luokan vastuulla on tarkastella syötteenä annettujen opetusnäytteiden ja luokitteluasetusten laillisuutta. F#-kielellä toteutettu *Classifier* toimii myös kääreenä MATLAB-komponentille, eli sen vastuulla on syötteiden käsitteleminen MATLAB-komponentille sopivaan muotoon, komponentin kutsuminen sekä tulosten käsittely ja niiden ohjaaminen kutsujalle.

Luokittelijan opettaminen tapahtuu kutsumalla rajapinnan *TrainClassifer*-metodia. Opetettava luokittelija voi olla tyypiltään SVM, RVM tai yksiluokkainen SVM ja kerrallaan voidaan käyttää vain yhtä luokittelija-algoritmia. SVM ja RVM ovat kaksiluokkaisia luokittelijoita, joten niitä voidaan käyttää silloin kun molemmista luokista on saatavilla opetusnäytteitä. Yksiluokkaista tukivektorikonetta puolestaan voidaan käyttää esimerkiksi tilanteessa, jossa toisesta luokasta ei ole saatavilla opetusnäytteitä. Muita käyttökohteita yksiluokkaiselle tukivektorikoneelle ovat tilanteet, joissa toisen luokan opetusnäytteitä on erityisen vähän verrattuna toiseen tai näytteet ovat kohinaisia tai muuten käyttökelvottomia.

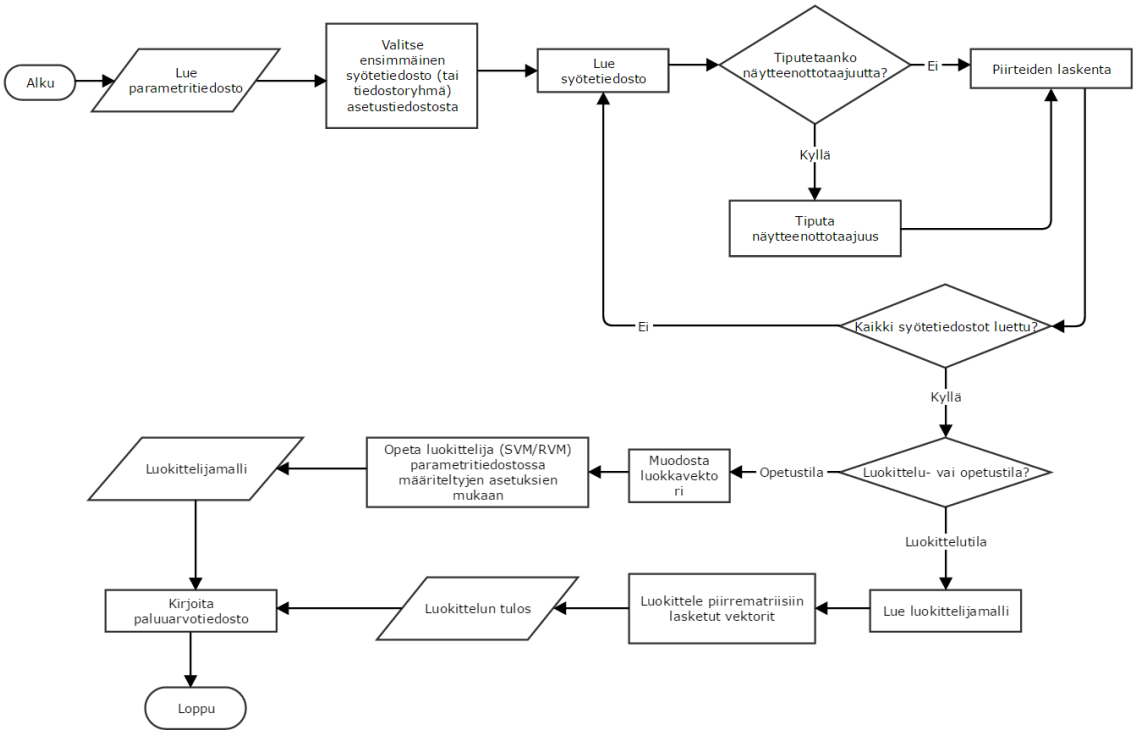
Luokittelijan syötteenä käytettäviä mittadatatiedostoja voidaan käsitellä kahdella tavalla: käyttämällä opetukseen jokaisesta mittadatatiedostosta erikseen laskettuja piirrevektoreita tai käsittelemällä mittadatatiedostoja ryhminä, jolloin ryhmän kullekin jäsenelle lasketaan piirrevektori, mutta ennen luokittelijan opettamista ryhmään kuuluvat piirrevektorit yhdistetään yhdeksi piirrevektoriksi laskemalla niistä keskiarvo tai mediaani. Tällä tavoin peräkkäisillä mittauskerroilla mitattuja näytteitä voidaan käsitellä yhtenä piirrematriisin rivinä, jossa mahdollisen mittauskertojen välisen hajonnan vaikutusta on pienennetty.

Metodin onnistuneen suorituksen jälkeen paluuarvona on luokittelijamalli mat-tiedostossa. MAT-tiedosto palautetaan metodin kutsujalle tiedostovirtana. Luokittelijamallia voidaan käyttää tuntemattomien näytteiden luokittelemiseen Classify-metodilla. Luokittelijamallin varastoiminen on isäntäjärjestelmän vastuulla. Myös opettamiseen käytetyt luokittelijan parametrit on syytä varastoida isäntäjärjestelmään, sillä tuntemattomia näytteitä luokiteltaessa piirteiden laskentaan tulee käyttää samoja parametreja kuin luokittelijaa opetettaessa. Tällöin luokiteltavasta näytteestä voidaan laskea samat piirteet kuin millä luokittelijamalli on opetettu.

Tuntemattomien näytteiden luokittelemiseen käytetään rajapinnan Classify-metodia, jonka parametreiksi annetaan luokitteluasetukset, kokoelma luokiteltavia näytteitä sekä luokittelijamalli, jota käyttäen näytteet luokitellaan. Onnistuneen suorituksen jälkeen paluuarvona saadaan luokittelun tulokset kokoelmana *ClassificationResult*-olioita. Mikäli käytettiin SVM-luokittelijaa, tuloksena on ainoastaan tieto siitä, mihin luokkaan kukin luokitelluista näytteistä kuuluu. Todennäköisyyttä kuvaavat arvot on asetettu nollaksi, sillä SVM-luokittelija ei tuota todennäköisyysarvoja luokittelun tuloksille. RVM-luokittelijaa käytettäessä luokittelijan määrittämän luokan lisäksi myös tulosten todennäköisyysarvot ovat käyttökelpoista dataa.

4.7 MATLAB-komponentti

MATLAB-komponentin vastuulla on kaikki vauriontunnistuskomponenttiin liittyvä signaalinkäsittely sekä luokittelijaan liittyvät toiminnot. Sen tehtäviä ovat näytteiden esikäsittely, piirteiden laskenta, luokittelijan opettaminen sekä näytteiden luokittelu luokittelijamallin perusteella. Komponentista käännetään MATLAB Compiler -työkalulla komentoriviltä ajettava ohjelma, jonka syötteenä on toiminnallisuutta ohjaava parametritiedoston. Parametritiedostossa on määritelty mm. syötetiedostot, laskettavat piirteet, käytettävä luokittelualgoritmi, luokittelualgoritmin konfiguraatioparametrit sekä se, käytetäänkö komponenttia luokittelu- vai opetustilassa. Komponentin toiminta on esitetty vuokaaviona kuvassa 4.2.



Kuva 4.2 *MATLAB-komponentin toiminta vuokaaviona.*

4.7.1 Näytteiden esikäsittely

Parametritiedostossa ryhmitellyistä näytteistä lasketaan konfiguraatiosta riippuen ensin mediaani tai keskiarvo ja tästä koosteesta koko ryhmälle muodostetaan yksi piirrevektori. Näin saadaan vähennettyä mahdollisen mittauskertojen välisen hajonnan vaikutusta luokittelijan toimintaan.

Mikäli luokittelun kannalta merkittävät taajuusalueet sijaitsevat näytteenottotaajuuden mahdollistaman maksimitaajuuskaistan alapäässä, laskennan tehostamiseksi mittadatan näytteenottotaajuutta voidaan alentaa ennen piirteiden laskemista. Tarpeettomia taajuuksia karsimalla myös kohinaa saadaan vähennettyä. Laskostumisen välttämiseksi näytteille tehdään alipäästösuodatus ennen näytteenottotaajuuden alentamista.

4.7.2 Piirteiden laskenta

Kaikista esikäsitellyistä näytteistä lasketaan parametritiedostossa määrätyt piirteet, joista muodostuu näytteelle piirrevektori. Jokaisesta näytteestä lasketaan samat piirteet ja ne asetellaan piirrevektoriin aina samassa järjestyksessä. Laskettavat piirteet on myös suunniteltava siten, että ne tuottavat samanmittaisen piirrevektorin näytteestä riippumatta.

Piirteet lasketaan aina näytteiden taajuustason esityksistä, eli tässä tapauksessa FFT-algoritmilla lasketuista Fourier-muunnoksista. Aikatasossa lasketut piirteet vaatisivat näytteiden keskinäistä synkronointia näytteen tarkkuudella, mikä ei ole mahdollista nykyisillä mittalaitteilla. Taajuustasossa laskettavat piirteet voidaan laskea joko näytteen koko taajuusalueelle tai ainoastaan erikseen määritellyille lyhyemmille taajuusalueille. Taajuusalueet voidaan määritellä esimerkiksi niin, että lasketaan piirre, jossa etsitään kolme taajuusvastefunktion huippukohtaa taajuusalueelta 0 - 50Hz, kaksi huippukohtaa alueelta 100 - 300Hz ja yksi huippukohta alueelta 320 - 350Hz.

Taajuusvastefunktion huippukohtien lisäksi piirteenä voidaan käyttää myös koko taajuusvastefunktiota tai taajuusaluein rajattuja osia siitä. Taajuusvastefunktiosta voidaan laskea piirteeksi myös joukko Rational Fraction Polynomial -menetelmällä [11] määritettyjä modaaliparametreja. Tällöin piirrevektoriin tallennetaan taajuusalueelta havaitut ominaistaajuudet, moodit ja taajuuksien vaimenemiskertoimet. Ominaistaajuuksien määrittäminen on laskennallisesti raskas operaatio paljon ominaistaajuuksia käsittävälle systeemille. MATLAB-komponentin kehittämiseen käytetyllä PC:llä (Intel Core i5-4570 @ 3.20Ghz, 16GB RAM) 150:n ominaistaajuuden määrittämiseen kului aikaa noin tunti. Laskentanopeuden parantamiseksi määritettävien modaaliparametrien määrää voidaan tiputtaa ja rajoittaa haku vain tietyille taajuusalueille.

Piirteiden laskennan tuloksena saadaan piirrematriisi, jonka rivit ovat näytteistä laskettuja piirrevektoreita ja sarakkeet ovat yksittäisiä piirteitä. Kuten alakohdassa 3.5.1 mainitaan, luokittelijan piirreavaruuden ulottuvuuksien määrä on sama kuin piirrevektorin sarakkeiden määrä. Piirrevektorin sarakkeiden määrän kasvaessa siis myös luokittelutehtävän kompleksisuus kasvaa. Kaikkein piirrematriisin sisältämien piirrevektoreiden tulee olla aina saman pituisia, jotta jokaisessa matriisin sarakkeessa on kaikilla riveillä samaa piirrettä kuvaava arvo.

4.7.3 Luokittelijan opettaminen

Luokittelijaa opetettaessa muodostetaan piirrematriisin pohjalta luokkavektori, jolla osoitetaan kunkin piirrematriisin rivin sisältämän näytteen luokka. Vektoriin tallennettavat arvot luetaan parametritiedostosta, jossa kullekin opetusnäytteelle tai opetusnäyteryhmälle on määritetty luokka. Vektori syötetään luokittelijan opetusfunktion parametrina yhdessä piirrevektorin kanssa. Opettamisen tuloksena syntyy luokittelijamalli.

Tukivektorikoneluokittelijan opettamisen yhteydessä piirrevektorista suodatetaan

epäoleellisia piirteitä tekemällä LibSVM-kirjaston tarjoamia työkaluja käyttäen ristivalidointi (cross-validation). Ristivalidoinnissa luokittelijan tarkkuutta optimoidaan etsimällä piirrematriisista merkitsevimmät piirteet opetusta varten. Ristivalidoinnissa LibSVM opettaa luokittelijan kaikilla mahdollisilla piirreyhdistelmillä ja vertailee, millä yhdistelmällä saavutetaan paras luokittelutarkkuus ja valitsee luokittelijamallissa käytettävät piirteet näiden tulosten mukaan. Opetus- ja testidatana käytetään opetus- ja testiosaan jaettua piirrematriisia, jossa kaikki näytteet sisältävästä piirrematriisista valitaan molempien luokkien edustajia kumpaankin joukkoon.

Muodostettu luokittelijamalli tallennetaan parametritiedostossa osoitettuun tiedostosijaintiin. Onnistuneen tallentamisen jälkeen MATLAB-komponentin suoritus loppuu. Komponenttia kutsunut .NET-kääre havaitsee tämän ja aloittaa MATLAB-komponentin tuottamien tulosten käsittelyn.

4.7.4 Luokittelu

Luokaltaan tuntemattomia näytteitä luokiteltaessa piirteiden laskennan jälkeen parametritiedostosta luetaan luokittelijamalli. Luokittelijamallia ja piirrematriisia käyttäen suoritetaan luokittelu valitun luokittelija-algoritmin toteutusta käyttäen. Luokittelussa kullekin näytteelle määritetään luokka ja RVM-luokittelijaa käytettäessä tuloksena on myös todennäköisyydet näytteen kuulumiselle eri luokkiin.

Luokittelun tulokset kirjoitetaan levylle parametritiedostossa määrättyyn tiedostosijaintiin. Tämän jälkeen Matlab-komponentin suorittaminen loppuu, jolloin .NET-kääre aloittaa luokittelutulosten tulkitsemisen.

5. MITTALAITEKOHTAINEN KALIBROINTI

Vauriontunnistuskomponentin luokittelijan opetukseen käytettävät värähtelymittaukset voivat olla peräisin erilaisista mittalaitteista. Eri mittalaitteista peräisin olevat mittaukset poikkeavat toisistaan huomattavasti, ja jo samantyyppisten mittalaitteiden tuottamien mittausten välillä voi olla merkittäviä eroja. Lisäksi samaa mittalaitteita voidaan käyttää kappaleen tuennan kannalta erilaisissa tilanteissa, mikä niin ikään vaikuttaa syntyneeseen mittaukseen. Mittalaitteen ja mittaustavan yhdistelmästä käytetään tässä termiä mittauskonfiguraatio.

Tarkan luokittelijamallin muodostamiseksi opetusdataa tarvitaan paljon. Koska eri mittauskonfiguraatioiden tuottamat värähtelymittaukset eivät ole keskenään vertailukelpoisia, jokaisesta mittalaitetyypistä ja jopa mittalaiteyksilöstä on tähän mennessä täytynyt mitata suuri kokoelma opetusdataa tarkan luokittelijamallin opettamiseksi. Syntynyttä luokittelijamallia on voitu käyttää ainoastaan tietyllä mittauskonfiguraatiolla mitattujen näytteiden luokitteluun.

Mittalaitekohtaisessa kalibroinnissa on tavoitteena saavuttaa tilanne, jossa eri mittauskonfiguraatioilla tuotettua mittausdataa on mahdollista käyttää keskenään luokittelijan opettamiseen. Tällöin voitaisiin opettaa luokittelijoita, joita voidaan käyttää millä tahansa näytteillä riippumatta siitä, millä mittalaitteella tai mittaustavalla ne on tuotettu.

5.1 Mittalaitetyypit

Kaikille käytössä oleville mittalaitteille on yhteistä, että niissä on herätevasaran kaltainen laite värähtelyn herättämiseksi sekä yksi tai useampia värähtelyä mittaavia antureita. Tässä keskitytään kahteen laitetyyppiin: mittausalustaan ja helpommin liikuteltavaan kappaleeseen kiinnitettävään mittalaitteeseen.

Mittausalustassa kiihtyvyysanturit on sijoitettu siten, että ne ovat kontaktissa mitattavan kappaleeseen kun kappale on asetettu alustan päälle. Värähtely herätetään mittausalustaan kiinnitetyllä herätevasaran kaltaisella laitteella, jonka laukaiseminen tapahtuu koneellisesti. Kiihtyvyysantureita alustassa voi olla laitemallista ja

konfiguraatiosta riippuen useita, mutta vähintään yksi heräte- ja yhden vastekanavan mittaamiseksi.

Kappaleeseen kiinnitettävä mittalaite puolestaan kiinnittyy mitattavaan kappaleeseen magneeteilla. Värähtely herätetään mittalaitteessa sisäänrakennettuna olevalla herätevasaraa vastaavalla laitteella. Heräte- ja vastesignaalien mittaaminen tapahtuu laitteeseen kiinnitetyillä kiihtyvyysantureilla. Mittalaitteessa on ainoastaan kaksi kanavaa: yksi herätteelle ja yksi vasteelle.

5.2 Näytteiden kalibroiminen

Kalibrointia varten määritetään referenssikappale, jota käyttäen mitataan värähtelymittaukset kaikilla sellaisilla mittauskonfiguraatioilla, joilla mitatuista näytteistä tullaan tekemään keskenään vertailukelpoisia kalibrointia käyttäen. Referenssikappaleen värähtelyominaisuuksien tulee pysyä vakiona mittauskertojen välillä vertailukelpoisuuden säilyttämiseksi. Jos siis referenssikappaleen värähtelyominaisuuksissa tapahtuu muutoksia esimerkiksi vahingoittumisen seurauksena, sitä ei voida enää käyttää tähän tarkoitukseen. Referenssikappaleen lisäksi määritellään jokin mittauskonfiguraatio, johon kaikki muut konfiguraatiot pyritään kalibraation avulla sovittamaan.

Kalibroinnin piiriin lisätään uusi mittauskonfiguraatio mittaamalla värähtelyä referenssikappaleesta kyseisellä konfiguraatiolla. Kalibroitavia kanavia on aina vähintään kaksi: heräte- ja vastekanavat. Mittausalustassa, jossa vastekanavia voi olla useita, kalibrointi tulee tehdä jokaiselle vastekanavalle erikseen. Mittauksia tehdään useita kohinan ja mittauskertojen välisen hajonnan vähentämiseksi. Jokaisesta näytteestä lasketaan diskreetti Fourier-muunnos FFT-algroritmilla ja näistä Fourier-muunnoksista lasketaan mediaani, jolloin tuloksena on yksi Fourier-muunnos.

Kalibrointidatan tuottamiseen tarvitaan referenssiksi valitulla mittauskonfiguraatiolla mitatun näytteen Fourier-muunnos. Referenssikonfiguraatiolla mitatun näytteen ja kalibroinnin piiriin lisättävällä konfiguraatiolla mitatun näytteen Fourier-muunnosten välinen suhde lasketaan. Koska FFT-algoritmin tuloksena on kompleksinen vektori, suhde tulee laskea reaali- ja imaginääriosalle erikseen. Kalibrointidata koostuu siis kahdesta vektorista: Fourier-muunnosten reaali- ja imaginääriosan suhdevektorista.

Näytteiden kalibroimiseksi tarvitaan näytteen mittaamiseen käytetylle mittauskonfiguraatiolle muodostettu kalibrointidata sekä näytteen Fourier-muunnos. Näytteen muunnosvektorin jokaisen alkion reaali- ja imaginääriosa kerrotaan kalibrointidatan

vastaavassa alkiossa olevilla reaali- ja imaginääriosien kertoimilla. Kertolaskun jälkeen näyte muutetaan takaisin aikatasoon käänteisellä Fast Fourier -muunnoksella (IFFT).

Kalibroituja näytteitä voidaan käyttää vauriontunnistuskomponentin syötteenä samaan tapaan kuin tavallisiakin näytteitä ja niistä voidaan myös laskea kaikki samat piirteet. Kalibroitujen näytteiden tallentaminen on isäntäjärjestelmän vastuulla. Isäntäjärjestelmään on näytteen lisäksi tallennettava myös tieto siitä, mihin mittauskonfiguraatioon näyte on kalibroitu. Mahdollisimman hyvän luokittelijamallin ja luokittelutulosten saavuttamiseksi luokittelijamallin opettamisessa tulisi käyttää ainoastaan yhteen mittauskonfiguraatioon kalibroituja näytteitä.

5.3 Kalibrointikomponentin toteutus

Näytteitä kalibroiva ohjelmistokomponentti toteutetaan vauriontunnistuskomponentista erillisenä moduulina. Tällöin isäntäjärjestelmässä voidaan kalibroida kaikki tarvittavat näytteet etukäteen ja tallentaa ne halutulla tavalla ilman, että luokittelijaa tarvitsisi opettaa samalla. Kuten vauriontunnistuskomponenttikin, kalibrointikomponentti integroidaan Microsoft .NET -sovelluskehyksellä toteutettuun isäntäjärjestelmään. Kalibrointikomponentin suhde isäntäjärjestelmään ja vauriontunnistuskomponenttiin on esitetty kuvassa 5.1.

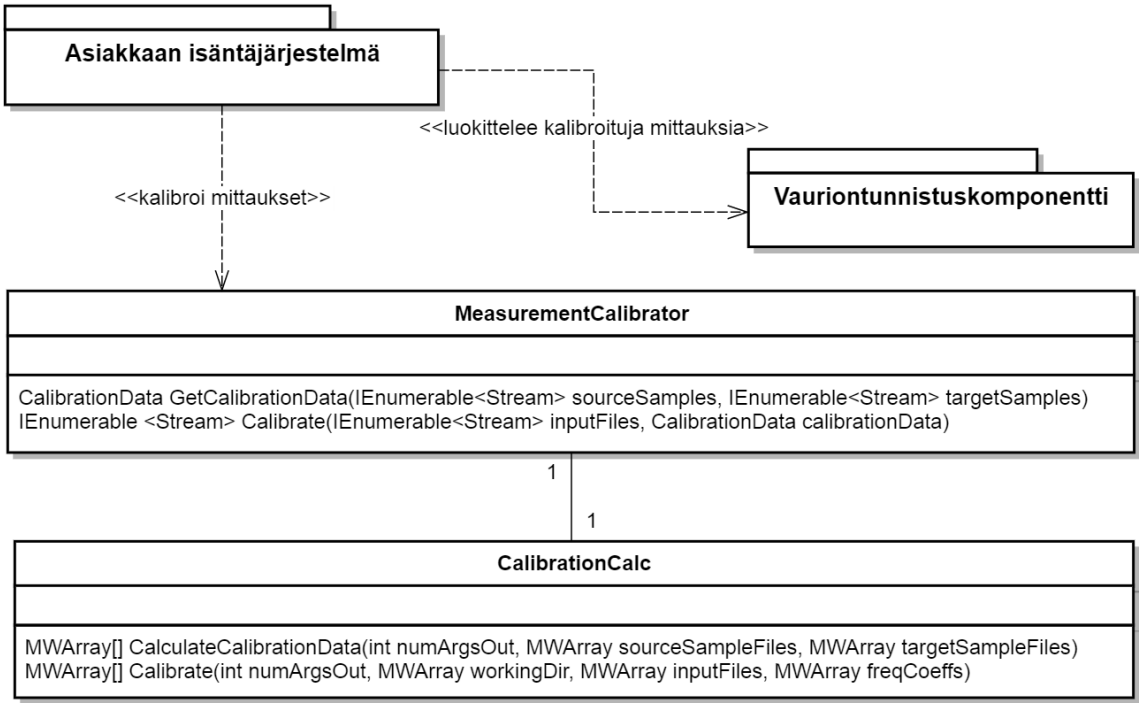
Kalibrointikomponenttiin liittyvä laskenta toteutetaan MATLAB-ympäristössä, joten arkkitehtuuri on hyvin samankaltainen vauriontunnistuskomponentin kanssa. Isäntäjärjestelmään integroitumista varten suunnitellaan kalibrointiin tarvittavat metodit tarjoava .NET-rajapinta. Rajapinnan toteuttava komponentti toimii kääreenä MATLAB-komponentille.

Kalibrointikomponentin toteutushetkellä MATLABin Compiler -työkalu oli päivitetty uusimpaan versioonsa, joka mahdollistaa MATLAB-koodin kääntämisen komentoriviohjelmien ja C++-kirjastojen lisäksi myös suoraan .NET-kirjastoksi. MATLABin tietotyyppejä voidaan käsitellä .NET-ympäristössä *MWArray*-kirjaston tarjoamilla työkaluilla.

MATLAB-komponentin ja kääreen välinen kommunikointi tapahtuu kalibrointikomponentissa suoraan .NET-funktiokutsujen avulla, eikä vauriontunnistuskomponentissa kommunikointiin käytettäviä levylle kirjoitettavia MAT-tiedostoja enää tarvita. Tämä vaikuttaa merkittävästi myös komponentin suorituskykyyn, sillä MATLAB Compilerin ajonaikainen kirjasto (MATLAB Compiler Runtime) ladataan ainoastaan ensimmäisellä kerralla MATLAB-komponentin funktioita kutsuttaessa ja

tämän jälkeen kirjasto pysyy ladattuna siihen asti kun MATLAB-komponenttia kutsuvan komponentin purkajaa kutsutaan. Vauriontunnistuskomponentin mukaisessa toteutuksessa ajonaikainen kirjasto ladataan uudestaan jokaisen MATLAB-komponentin kutsun yhteydessä.

Kuvassa 5.1 on esitetty kalibrointikomponentin arkkitehtuuri. *MeasurementCalibrator* tarjoaa isäntäjärjestelmälle metodit näytteiden kalibroimiseen sekä kalibrointidatan muodostamiseen referenssikappaleella mitatuista näytteistä. Komponentin tehtävinä on tarkistaa syötteiden oikeellisuus, käsitellä syötteet MATLAB-komponentille sopivaksi sekä muuttaa MATLAB-komponentin tulokset isäntäjärjestelmälle sopivaan muotoon. Kääre myös kutsuu MATLAB-komponenttia (*CalibrationCalc*).



Kuva 5.1 *Kalibrointikomponentin arkkitehtuuri ja suhde muihin järjestelmän osiin.*

Rajapinnan *GetCalibrationData*-metodia käyttäen saadaan laskettua kalibrointidata. Sen syötteinä ovat kalibraation piiriin lisättävällä mittauskonfiguraatiolla mitattuja näytteitä referenssikappaleesta (*sourceSamples*). Lisäksi syötteenä on referenssikonfiguraatiolla mitattuja näytteitä (*targetSamples*). Näytteet kalibroidaan *Calibrate*-metodilla. Syötteenä on kalibroitavat tiedostot (*inputFiles*) sekä kalibroinnissa käytettävä kalibrointidata (*calibrationData*). Paluuarvona on metodille syötetyt näytteet kalibroituina.

Kuten arkkitehtuurikuvasta havaitaan, MATLAB-komponentin kaikki syötteet ja

ulostulot ovat *MWArray*-olioita. *MWArray* on MATLABin tietotyyppien kantaluokka, josta muut MATLABin tietotyyppit perityvät [42]. Kääre muuttaa syötteet MWArray-kantaluokasta periytyviksi olioiksi ja kutsuu niitä parametreina käyttäen soveltuvaa MATLAB-komponentin metodia.

MATLAB-komponentin *CalculateCalibrationData*-metodilla lasketaan kalibrointidata edellä kuvatulla menetelmällä. Metodin paluuarvona on kalibroitavan kappaleen ja referenssikappaleen suhdetta kuvaavan kerroinvektorin reaali- ja imaginääriosat erillisissä vektoreissa. *Calibrate*-metodilla puolestaan kalibroidaan syötteenä annetut näytteet.

6. ARVIOINTI JA JATKOKEHITYS

Tässä luvussa arvioidaan vauriontunnistuskomponentin sekä mittalaitekohtaisen kalibroinnin toteutuksien onnistuneisuutta sekä komponenttien soveltuvuutta tehtäviinsä. Lisäksi esitetään joitain ehdotuksia komponenttien kehittämiseksi tulevaisuudessa.

Kohdat 6.1 ja 6.2 keskittyvät vauriontunnistuskomponentin arviontiin ja siihen liittyvien jatkokehitysehdotuksien esittämiseen. Komponenttia arvioidaan sille asetettujen vaatimusten, suorituskyvyn ja asiakaskokemusten perusteella. Kohdassa 6.3 puolestaan arvioidaan kalibrointimenetelmää ja esitetään sille jatkokehitysajatuksia.

6.1 Vauriontunnistuskomponentin arviointi

Tärkeimpänä vauriontunnistuskomponentille asetettuna tavoitteena oli kehittää ohjelmisto, joka täyttää asiakasyrityksen tarpeen kyetä havaitsemaan vaurioita kappaleista värähtelymittausten perusteella. Vaatimuksena oli myös, että toteutettava ohjelmisto kykenee kunnoltaan tunnetuista kohteista mitattujen värähtelymittausten perusteella muodostamaan luokittelijan, jolla voidaan luokitella kunnoltaan tuntemattomista kohteista mitattuja näytteitä.

Näiden vaatimusten perusteella vauriontunnistuskomponenttissa käytettäviksi valittiin koneoppimisalgoritmit, jotka perustuvat ohjattuun oppimiseen. Toteutukseen valitut SVM- ja RVM-luokittelijat suoriutuvat tehtävästä hyvin. Vauriontunnistuskomponentin laajan konfiguroitavuuden vuoksi luokittelun onnistumiseen vaikuttaa oleellisesti käyttäjän määrittämä luokittelijan konfiguraatio, joka käsittää mm. käytettävän luokittelualgoritmin, kernelfunktiot sekä piirrematriisiin sisällytettävät piirteet.

Yhtenä vaatimuksena oli, että vauriontunnistuskomponentin tulee olla integroitavissa Microsoft .NET -sovelluskehyksellä toteutettuun isäntäjärjestelmään. Vaatimuksen täyttämiseksi määriteltiin asiakasyrityksen ohjelmistosuunnittelijoiden kanssa yhteistyössä .NET-rajapinta. Lisäksi toteutettiin rajapinnan toiminnot toteuttava

kääre, joka kätkee isäntäjärjestelmältä MATLAB-komponentin ja kääreen välisen kommunikaation. Rajapinta on sovellusalueen monimutkaisuudesta huolimatta yksinkertainen, joskin sen käyttäjältä vaaditaan jonkin verran tietämystä saatavilla olevista piirteistä sekä luokittelualgoritmeihin liittyvästä käsitteistöstä.

Vauriontunnistuskomponentin toteutushetkellä MATLAB-koodin kääntämiseen tarkoitettu MATLAB Compiler -työkalu mahdollisti ainoastaan C++-kirjastojen ja komentoriviltä ajettavien ohjelmien kääntämisen. MATLAB Compilerilla tuotetun C++-kirjaston integroimista .NET-kääreen kanssa kokeiltiin, mutta kaikkia luokittelussa tarvittavia tietotyyppejä ei pystytty sen kautta välittämään. Näiden rajoitteiden puitteissa MATLAB-komponentti päädyttiin kääntämään komentoriviltä ajettavaksi ohjelmaksi, ja samalla kääreen ja MATLAB-komponentin välinen kommunikointi katsottiin järkevimmäksi toteuttaa levylle kirjoitettavilla MAT-tiedostoilla.

Komponenttien välisen kommunikaation toteutus on levylle kirjoittamisen vuoksi hidasta verrattuna siihen, että kommunikointi tapahtuisi keskusmuistissa säilytettävien objektien välityksellä. Eniten suorituskykyyn vaikuttaa kuitenkin MATLABin ajonaikainen kirjasto Matlab Compiler Runtime, joka joudutaan lataamaan uudestaan jokaisella MATLAB-komponentin kutsulla. Ajonaikaisen kirjaston lataaminen kestää useita sekunteja, joten vaikutus suorituskykyyn on merkittävä.

Luokittelijaa opetettaessa kirjaston lataamisen vaikutus suorituskykyyn ei ole oleellinen tekijä, sillä luokittelijamallin muodostaminen kestää muutenkin kauan silloin kun opetusnäytteitä on paljon. Lisäksi uusia luokittelijamalleja luodaan verraten harvoin. Kunnoltaan tuntemattomien näytteiden luokittelun tulisi kuitenkin tapahtua nopeasti, jolloin myös ajonaikaisen kirjaston lataamisen aiheuttama suorituskyvyn heikkeneminen saattaa olla merkityksellistä. Huolimatta kirjaston lataamisen vaikutuksista suorituskykyyn, asiakasyritykseltä saadun palautteen perusteella vauriontunnistuskomponentin suorituskyky on kuitenkin ollut nykytoteutuksessaankin riittävää.

Komponentin suorituskykyongelmista huolimatta luokittelijan ja piirteiden laskennan toteuttaminen MATLAB-koodina osoittautui hyväksi valinnaksi. Luokittelijan syötteenä käytettävien piirrevektorien ja -matriisien muodostaminen ja käsittely on matriisilaskentaan tarkoitetulla MATLABilla vaivatonta. MATLABissa myös tiedon visualisointi onnistuu helposti, mikä osoittautui komponentin kehitysvaiheessa tärkeäksi ominaisuudeksi. Piirteiden kehittäminen oli helpompaa, kun mitatun kappaleen vaurioiden vaikutuksia esimerkiksi taajuusvastefunktiossa voitiin tarkastella visuaalisesti.

Vauriontunnistuskomponentista kehitettiin yleiskäyttöinen työkalu, jota voidaan käyt-

tää värähtelymittauksien luokitteluun käytetystä mittalaitteistosta riippumatta. Tässä onnistuttiin hyvin, sillä vauriontunnistuskomponentti ei ota kantaa mittalaitteeseen, jolla sille syötettäviä näytteitä on mitattu. Eri mittauskonfiguraatioilla mitattujen näytteiden käyttäminen saman luokittelijan opettamiseen sen sijaan ei ole mahdollista ilman kaikkien mittauskonfiguraatioiden kalibrointia. Vauriontunnistuskomponentissa tehtävä näytetiedostojen signaalinkäsittely on myös toteutettu niin, että syötetiedostojen näytteenottotaajuus tai bittisyvyys ei vaikuta sen toimintaan muuten kuin laskenta-ajan osalta.

Asiakasyritykseltä saadun palautteen perusteella vauriontunnistuskomponentin luokittelijan tarkkuus on nykytoteutuksessa riittävä. Luokittelijan tarkkuuteen vaikuttaa merkittävästi käytetyt piirteet sekä luokittelija-algoritmin konfiguraatio. Oletettavasti parhaan mahdollisen luokittelutarkkuuden saavuttaminen on vaatinut paljon kokeiluja erilaisilla piirreyhdistelmillä ja luokittelijakonfiguraatioilla.

6.2 Vauriontunnistuskomponentin jatkokehitys

Kalibrointikomponentin kehittämisen yhteydessä havaittiin, että MATLABin nykyversioissa Compiler-lisäosan tarjoamaa toiminnallisuutta on laajennettu siten, että sillä voidaan kääntää myös .NET-kirjastoja. MATLAB-komponentti voitaisiin siis nykyisillä työkaluilla integroida suoraan kääreeseen eikä sitä tarvitsisi enää kääntää komentorivityökaluksi. Samalla viestinvälitys voitaisiin toteuttaa täysin .NET-rajapinnan kautta ja luopua levylle kirjoitettavien MAT-tiedostojen käyttämisestä kokonaan. Tämä parantaisi myös komponentin suorituskykyä, sillä hitaasta levylle kirjoittamisesta voitaisiin luopua. Lisäksi nykytoteutuksessa suorituskykyä merkittävästi heikentävä MATLABin ajonaikaisen kirjaston lataaminen tarvitsisi tehdä ainoastaan yhden kerran.

Vauriontunnistuskomponentin monikäyttöisyyttä voidaan entisestään parantaa mahdollistamalla useamman kuin kahden luokan luokittelijoiden käyttäminen. Tässä vaiheessa komponentin kehitystä tavoitteena oli pystyä luokittelemaan kappaleista mitattuja näytteitä ehjiin ja rikkinäisiin, joten binääristen luokittelijoiden käyttäminen oli perusteltua. Mikäli vauriontunnistusta halutaan jatkossa kehittää tunnistamaan esimerkiksi erilaisia vauriotyyppejä, voidaan tässä kenties hyödyntää moniluokkaisia luokittelijoita.

Vauriontunnistuskomponentin tukivektorikonetoteutuksena käytettävä LibSVM-kirjasto tukee moniluokkaisia tukivektorikoneita, mutta vauriontunnistuskomponenttiin ei vielä ole toteutettu tukea sellaisille. Valittua RVM-toteutusta ei suoraan voida käyttää moniluokkaisiin luokittelutehtäviin, mutta moniluokkaisuuden tuki voidaan

kehittää siihen tarvittaessa itse.

Vaikka vauriontunnistuskomponenttiin toteutetut piirteet ovat käyttökelpoisia, uusien piirteiden kehittäminen ja helppo testaaminen vauriontunnistuskomponentin luokittelijoilla voi tulla kyseeseen. Nykytoteutuksessa piirteitä voidaan lisätä, muuttaa tai poistaa ainoastaan lähdekoodia muokkaamalla ja kääntämällä sitten uusi versio MATLAB-komponentista.

Piirteiden lisääminen uutta versiota kääntämättä voitaisiin mahdollistaa erottamalla vauriontunnistuskomponenttiin toteutetut piirteiden laskenta ja luokittelijan toimintojen käyttäminen erillisiksi metodeikseen rajapinnassa. Tällöin näytteistä voitaisiin muodostaa luokittelijalle syötettäviä piirrevektoreita käyttämällä vauriontunnistuskomponentin tarjoamia valmiita piirteenlaskenta-algoritmeja, eli lähes samalla tavalla kuin nykytoteutuksessa.

Valmiiden piirteenlaskentametodien käyttämisen lisäksi piirrevektorien muodostaminen olisi tässä arkkitehtuurissa mahdollista myös siten, että piirrevektorit muodostettaisiin vauriontunnistuskomponentista erillään toteutetuista piirrealgoritmeilla. Vauriontunnistuskomponentin käyttäminen isäntäjärjestelmästä vaatisi nykytoteutukseen verrattuna yhden metodin kutsumisen enemmän, mutta toisaalta uusia piirteitä voitaisiin tällöin kehittää ja testata luokittelijalla merkittävästi nykytoteutusta yksinkertaisemmin ja vauriontunnistuskomponentin lähdekoodia muuttamatta.

Vauriontunnistuskomponentissa käytettäviksi luokittelija-algoritmeiksi valittiin tukivektorikone ja relevanssivektorikone. Valittujen algoritmien lisäksi ominaisuuksiltaan soveltuvaksi algoritmiksi havaittiin myös monikerrosperseptroni (MLP), mutta se jätettiin potentiaalisesti hitaamman luokittelunopeutensa vuoksi tässä vaiheessa toteutuksen ulkopuolelle. Tulevaisuudessa myös MLP:n käyttömahdollisuuksia vauriontunnistuskomponenttissa voidaan tutkia esimerkiksi vertailemalla sen tuottamien luokittelutuloksien tarkkuutta nykyisten algoritmien tarkkuuteen.

6.3 Mittalaitekohtaisen kalibroinnin arviointi ja jatkokehitys

Mittalaitekohtaisessa kalibroinnissa tavoitteena oli saavuttaa tilanne, jossa eri mittauskonfiguraatioilla tuotettua mittausdataa on mahdollista käyttää keskenään luokittelijan opettamiseen. Tässä työssä kuvatulla taajuuksien korjaamiseen perustuvalla kalibrointimenetelmällä tämä on osittain mahdollista, mutta menetelmä on vielä huomattavan puutteellinen. Tällä hetkellä kalibrointi soveltuu käytettäväksi kahden samantyyppisen mittalaiteyksilön tuottamien mittausten yhdenmukaistamiseen

silloin kun käytetään samaa mittaustapaa. Kalibroinnin nykytoteutus on käytännössä monikaistainen FFT-taajuuskorjain, eikä se ota huomioon kappaleen tuennan aiheuttamia muutoksia kappaleen värähtelyominaisuuksiin. Samantyyppisten mittalaiteyksilöiden tuottamien mittausten välilläkin on merkittäviä eroavaisuuksia, joten tällaisten näytteiden kalibrointimahdollisuutta voidaan kuitenkin pitää edistysaskeleena kohti täysimittaisempaa kalibrointia.

Kalibrointia kehitettäessä havaittiin joidenkin mittauskonfiguraatioiden tuottavan säröytyneitä mittauksia kiihtyvyysantureiden signaalin yliohjautumisen vuoksi. Asiakasyrityksen mukaan ilmiötä ei voida välttää ja säröytyneiden näytteiden kalibrointi tulisi olla mahdollista. Tällaisten signaalien perusteella kalibrointia ei kuitenkaan nykytoteutuksella voida luotettavasti suorittaa. Säröytymisen yhteydessä tuhoutunutta tietoa ei voida koskaan palauttaa täysin alkuperäiseksi, mutta kalibrointikomponenttiin voidaan tulevaisuudessa yrittää löytää menetelmä, jolla leikkautuneesta värähtelydatasta saadaan jäljiteltyä mahdollisimman paljon alkuperäistä värähtelyä vastaava.

7. YHTEENVETO

Tässä diplomityössä kuvattiin värähtelyperustaisen vauriontunnistuskomponentin suunnittelu ja toteutus. Tavoitteena oli kehittää vauriontunnistuskomponentti, joka kykenee määrittämään värähtelymittauksien perusteella, onko analysoitava kappale ehjä vai vaurioitunut. Lisäksi esitettiin menetelmä, jolla erilaisilla mittalaitteilla ja mittaustavoilla mitatut näytteet voidaan kalibroida keskenään vertailukelpoisiksi.

Vauriontunnistuskomponentille määriteltiin yksinkertainen .NET-rajapinta, joka tarjoaa metodit luokittelijan opettamiseen ja kunnoltaan tuntemattomien näytteiden luokittelemiseen. Varsinainen laskenta, eli luokittelijalle syötettävien piirteiden määrittäminen, luokittelijan opettaminen ja tuntemattomien näytteiden luokittelu, toteutettiin MathWorksin MATLAB-laskentaympäristössä. Isäntäjärjestelmään integroituvan .NET-rajapinnan toteuttava F#-ohjelmointikielellä toteutettu komponentti toimii kääreenä MATLAB-komponentille. Toisin sanoen se kätkee isäntäjärjestelmältä MATLAB-komponentin kutsumisen, syötteiden käsittelyn ja tulosten tulkitsemisen.

Vauriontunnistuskomponentti suunniteltiin siten, että sitä voidaan käyttää yleiskäyttöisenä työkaluna kappaleiden luokittelemiseen värähtelymittausten perusteella. Se ei ole riippuvainen mittauksiin käytetystä mittalaitteesta, kunhan mittalaitteen tuottamat näytteet voidaan muuttaa WAV-muotoisiksi aaltoäänitiedostoiksi.

Erilaisilla mittauslaitteilla ja -tavoilla mitattujen näytteiden kalibrointi ei vielä nykymuodossaan täytä kaikkia sille asetettuja odotuksia. Tässä työssä kehitetyllä menetelmällä voidaan kalibroida kahdella samantyyppisellä mittalaitteella mitatut näytteet siten, että niitä voidaan käyttää keskenään luokittelijan opettamiseen ja näytteiden luokittelemiseen.

Tulevaisuudessa vauriontunnistuskomponenttia voidaan kehittää toteuttamalla tuki moniluokkaisille luokittelijoille, jolloin vaurioiden tunnistamisessa voidaan saavuttaa nykyistä jaottelua hienojakoisempi vaurioiden analysointimenetelmä. Vauriontunnistuskomponentin osien välistä kommunikaatiota ja suorituskykyä voidaan parantaa päivitetyn Matlab Compiler -version mahdollistamalla suoralla .NET-in-

tegraatiolla. Uusien piirteiden lisäämiseksi voidaan tulevaisuudessa vauriontunnistuskomponentin piirteiden laskenta ja luokittelutoiminnot erottaa toisistaan siten, että luokittelijan syötteenä voidaan käyttää myös vauriontunnistuskomponentin ulkopuolella näytteistä laskettuja piirteitä.

LÄHTEET

[1] K. D. Morton, P. Torrione, L. Collins, S. Keene, An Open Source Pattern Recognition Toolbox for MATLAB, Covartech, 2014, 6 s. Saatavissa (haettu 8.4.2016): http://arxiv.org/pdf/1406.5565v1.pdf.

[2] D. C. Zimmerman, T. Simmermacher, M. Kaouk, Structural Damage Detection Using Frequency Response Functions, 1995, pp. 179–184.

[3] M. Willcox, G. Downes, A Brief Description of NDT Techniques, Insight NDT, 2000, 22 s. Saatavissa (haettu 15.4.2016): http://www.turkndt.org/sub/makale/ornek/a%20brief%20description%20of%20ndt.pdf.

[4] D. J. Ewins, *Modal testing : theory, practice and application, 2nd edition*, Research Studies Press, Baldock, England, 2000, 562 s.

[5] *The Fundamentals of Modal Testing.* Application Note 243-3, Agilent Technologies, 2000, 56 s. Saatavissa (haettu 15.4.2016): http://www.modalshop.com/techlibrary/Fundamentals%20of%20Modal%20Testing.pdf.

[6] Brüel & Kjaer -yrityksen verkkosivusto, 2016. Saatavissa (haettu 8.4.2016): http://www.bksv.com/Products.

[7] W. A. Fladung, Windows used for impact testing, in *In Proceedings of the 15th International Modal Analysis Conference*, 1997, pp. 1662–1666.

[8] D. J. Inman, *Engineering Vibration, 4th edition.* Pearson Education, USA, 2013, 720 s.

[9] C. W. de Silva, *Vibration: Fundamentals and Practice, Second Edition.* CRC Press LLC, 2006, 1064 s.

[10] E. O. Brigham, *The fast Fourier transform.* Prentice-Hall signal processing series, Prentice-Hall, Englewood Cliffs, New Jersey, 1974, 252 s.

[11] M. H. Richardson, D. L. Formenti, Parameter estimation from frequency response measurements using rational fraction polynomials, *In Proceedings of the 1st International Modal Analysis Conference*, 1982, pp. 167–181.

[12] J. F. Hauer, C. J. Demeure, L. L. Scharf, Initial results in Prony analysis of power system response signals, *IEEE Transactions on Power Systems*, vol. 5, 1990, pp. 80–89.

[13] M. A. Mannan, M. H. Richardson, Detection and location of structural cracks using FRF measurements, Royal Institute of Technology, Stockholm, Sweden, 1990, 6 s.

[14] M. Mohri, A. Rostamizadeh, A. Talwalkar, *Foundations of Machine Learning.* The MIT Press, 2012, 432 s.

[15] C. M. Bishop, *Pattern recognition and machine learning*, Springer, 2006, 738 s.

[16] B. E. Boser, I. M. Guyon, V. Vapnik, A Training Algorithm for Optimal Margin Classifiers, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, ACM Press, 1992, pp. 144–152.

[17] M. E. Tipping, A. Smola, Sparse Bayesian Learning and the Relevance Vector Machine, in *Journal of Machine Learning Research 1*, 2001, pp. 211–244. Saatavissa (haettu 8.4.2016): `http://www.jmlr.org/papers/volume1/tipping01a/tipping01a.pdf`.

[18] S. Haykin, *Neural Networks: A Comprehensive Foundation, 2nd edition*, Prentice Hall, 1998, 842 s.

[19] T. Kohonen, The self-organizing map, *Proceedings of the IEEE*, vol. 78, 1990, pp. 1464–1480.

[20] R. S. Sutton, A. G. Barto, *Introduction to Reinforcement Learning, 2nd edition*, MIT Press, London, England, 1998, 320 s.

[21] E. Frias-Martinez, A. Sanchez, J. Velez, Support vector machines versus multi-layer perceptrons for efficient off-line signature recognition, *Engineering Applications of Artificial Intelligence*, vol. 19, no. 6, 2006, pp. 693 – 704. Saatavissa (haettu 22.4.2016): `http://www.sciencedirect.com/science/article/pii/S0952197606000352`.

[22] T. Hofmann, B. Schölkopf, A. J. Smola, Kernel methods in machine learning, *The Annals of Statistics*, vol. 36, 2008, pp. 1171–1220. Saatavissa (haettu 8.4.2016): `http://arxiv.org/pdf/math/0701907.pdf`.

[23] L. Wang, *Support Vector Machines: Theory and Applications.* Studies in Fuzziness and Soft Computing, Springer, Berlin, 2005, 431 s.

[24] J. Hu, P. W. Tse, A Relevance Vector Machine-Based Approach with Application to Oil Sand Pump Prognostics, *Sensors*, vol. 13, no. 9, 2013, pp. 12663–12686. Saatavissa (haettu 8.4.2016): `http://www.mdpi.com/1424-8220/13/9/12663`.

[25] N. C. Naveen, S. Natarajan, R. Srinivasan, Application of Relevance Vector Machines in Real Time Intrusion Detection, *International Journal of Advanced Computer Science and Applications(IJACSA)*, vol. 3, no. 9, 2012, pp. 48–53. Saatavissa (haettu 21.4.2016): `http://ijacsa.thesai.org/`.

[26] M. A. Hall, L. A. Smith, Practical Feature Subset Selection for Machine Learning, Department of Computer Science, University of Waikato, Hamilton, New Zealand, 1998, 11 s.

[27] H. Liu, H. Motoda, *Computational Methods of Feature Selection (Chapman & Hall/Crc Data Mining and Knowledge Discovery Series)*, Chapman & Hall/CRC, 2007, 440 s.

[28] I. Guyon, S. Gunn, M. Nikravesh, L. A. Zadeh, *Feature Extraction: Foundations and Applications (Studies in Fuzziness and Soft Computing)*, Springer, New York, USA, 2006, 778 s.

[29] J. Shawe-Taylor, N. Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press, New York, USA, 2004, 478 s.

[30] B. Schölkopf, J. C. Platt *et al.*, Estimating the Support of a High-Dimensional Distribution, *Neural Computation*, vol. 13, 2001, pp. 1443–1471.

[31] D. G. Tzikas, L. Wei *et al.*, A tutorial on relevance vector machines for regression and classification with applications, Department of Computer Science, University of Ioannina, Ioannina, Greece, 25 s. Saatavissa (haettu 5.3.2016): `http://www.ssp.ece.upatras.gr/galatsanos/PAPERS/IPL_papers/RVM_tutorial.pdf`.

[32] C.-C. Chang, C.-J. Lin, LIBSVM: A library for support vector machines, *ACM Transactions on Intelligent Systems and Technology*, vol. 2, 2011, pp. 27:1–27:27. Saatavissa (haettu 8.4.2016): `https://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf`.

[33] The BSD 3-Clause License, Regents of the University of California, 1999, Saatavissa (haettu 8.4.2016): `https://opensource.org/licenses/BSD-3-Clause`.

[34] Statistics and Machine Learning Toolbox -verkkosivu, 2016, Saatavissa (haettu 8.4.2016): `http://se.mathworks.com/help/stats/index.html`.

[35] The MIT License, Massachusetts Institute of Technology, 1988, Saatavissa (haettu 18.3.2016): `https://opensource.org/licenses/mit-license.php`.

[36] M. E. Tipping, An Efficient MATLAB Implementation of the Sparse Bayesian Modelling Algorithm (Version 2.0), 2009, 15 s.

[37] M. E. Tipping, SPARSEBAYES V1.1: A Baseline MATLAB Implementation of "Sparse Bayesian"Model Estimation, 2009, 5 s.

[38] GNU General Public License, Free Software Foundation, 1991, Saatavissa (haettu 8.4.2016): `http://www.gnu.org/licenses/old-licenses/gpl-2.0.html`.

[39] D. E. King, Dlib-ml: A Machine Learning Toolkit, *Journal of Machine Learning Research*, vol. 10, 2009, pp. 1755–1758. Saatavissa (haettu 8.4.2016): `http://jmlr.csail.mit.edu/papers/volume10/king09a/king09a.pdf`.

[40] MathWorks, MATLAB Runtime, 2016, Saatavissa (haettu 15.4.2016): `http://se.mathworks.com/products/compiler/mcr/`.

[41] H. Huttunen, Signaalinkäsittelyn perusteet luentomoniste, Signaalinkäsittelyn laitos, Tampereen teknillinen yliopisto, 2014, 151 s, Saatavissa (haettu 15.4.2016): `http://www.cs.tut.fi/kurssit/SGN-11000/SGN-11000.pdf`.

[42] *MATLAB Compiler SDK .NET User's Guide*, MathWorks Inc., 2015, 280 s.